ORIGINAL

Walker & Jocke

a legal professional association

Ralph E. Jocke
Patent
&
Trademark Law

3621

June 18, 2002

Board of Patent Appeals and Interferences
Commissioner for Patents
Washington, D.C. 20231

Re:	**Application Serial No.:**	**09/193,787**
	Confirmation No.:	**2446**
	Appellants:	**Jay Paul Drummond, et al.**
	Title:	**Automated Banking Machine Apparatus and System**
	Docket No.:	**D-1077+1**

Sir:

Please find enclosed the Brief of Appellants pursuant to 37 C.F.R. § 1.192 in triplicate for filing in the above-referenced application.

It is believed that no extension of time is required. However, if such an extension is required then please consider this a petition therefor.

Please charge the fee required with this filing ($320) and any other fee due to Deposit Account 09-0428.

Very truly yours,

Ralph E. Jocke
Reg. No. 31,029

CERTIFICATE OF MAILING BY EXPRESS MAIL

I hereby certify that this document and the documents indicated as enclosed herewith are being deposited with the U.S. Postal Service as Express Mail Post Office to addressee in an envelope addressed to Board of Patent Appeals and Interferences, Commissioner for Patents, Washington, D.C. 20231 this 20th day of June 2002.

EV044451383US
Express Mail Label No.

Ralph E. Jocke

330 • 721 • 0000 MEDINA
330 • 225 • 1669 CLEVELAND
330 • 722 • 6446 FACSIMILE
rej@walkerandjocke.com E-MAIL
231 South Broadway, Medina, Ohio U.S.A. 44256-2601

120 — 320.02 AF 7-2702 mee

D-1077+1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of:	**Jay Paul Drummond, et al.**	Art Unit 2161
Serial No.:	**09/193,787**	Patent Examiner Jalatee Worjloh
Filed:	**November 17, 1998**	
Title:	**Automated Banking Machine Apparatus and System**	

Board of Patent Appeals and Interferences
Commissioner for Patents
Washington, D.C. 20231

BRIEF OF APPELLANTS PURSUANT TO 37 C.F.R. § 1.192

Sir:

The Appellants hereby submit their Brief pursuant to 37 C.F.R. § 1.192, in triplicate, concerning the above-referenced Application.

REAL PARTY IN INTEREST

The Assignee of all right, title and interest to the above-referenced Application is Diebold, Incorporated, an Ohio corporation.

RELATED APPEALS AND INTERFERENCES

Appellants believe that there are no related appeals or interferences pertaining to this matter.

STATUS OF CLAIMS

Claims 1-30 are pending in the Application.

Claim 27 was rejected pursuant to 35 U.S.C. § 112, second paragraph.

Claims 1-6, 8-13, 16, and 22-26 were rejected pursuant to 35 U.S.C. § 102(e) as being anticipated by Zeanah et al. (hereinafter "Zeanah").

Claims 7, 14, 28, and 30 were rejected pursuant to 35 U.S.C. § 103(a) as being unpatentable over Zeanah in view of Russell et al. (hereinafter "Russell").

Claim 29 was rejected under 35 U.S.C. § 103(a) as being unpatentable over Zeanah in view of Russell and McLean.

Claims 15 and 17 were rejected pursuant to 35 U.S.C. § 103(a) as being unpatentable over Zeanah in view of McLean.

Claims 18-21 were rejected pursuant to 35 U.S.C. § 103(a) as being unpatentable over Zeanah in view of McMillan.

These rejections were the only rejections present in the Office Action ("Action") dated April 16, 2002. Appellants appeal the rejections of all claims.

STATUS OF AMENDMENTS

No final rejection has been made. However, claims have been twice rejected. Therefore, no amendments to the claims were requested to be admitted after a final rejection.

SUMMARY OF INVENTION

Overview of the Invention

An exemplary embodiment of the present invention is directed to an apparatus including an Automated Teller Machine (ATM) (12). The ATM includes an output device, an input device, a transaction function device, and a computer. A user of the ATM is able to perceive outputs from the output device (e.g., display screen) during communication with the ATM. The input device (e.g., touch screen (30)) is operative to receive user input from a user of the ATM. The transaction function device (36) may operate to carry out banking transaction functions. The transaction function device may comprise a cash dispenser (42) capable of dispensing cash. The computer (34) is in operative connection with the output device, input device, and transaction function device.

Software is executable in the computer. The software includes a browser (76) that is operative to process HTML documents. The HTML documents may include instructions adapted to cause operation of the transaction function device. The ATM operates the transaction function device in response to the instructions. Thus, a transaction function device is operated to carry out the banking transaction function responsive to the processing of an HTML document having

instructions. The computer is also operative to display a visual output through the display screen responsive to the processing of document instructions. Thus, an ATM of the exemplary embodiment is operative to receive HTML documents having instructions, process the document instructions, and provide both an output through the display screen and the dispense of cash responsive to the instructions.

CONCISE STATEMENT OF THE ISSUES PRESENTED FOR REVIEW

The questions presented in this appeal are:

1). Whether Appellants' claim 27 is unpatentable under 35 U.S.C. § 112, second paragraph.

2). Whether Appellants' claims 1-6, 8-13, 16, and 22-26 are unpatentable under 35 U.S.C. § 102(e) as being anticipated by Zeanah.

3). Whether Appellants' claims 7, 14, 28, and 30 are unpatentable under 35 U.S.C. § 103(a) over Zeanah in view of Russell.

4). Whether Appellants' claim 29 is unpatentable under 35 U.S.C. § 103(a) over Zeanah in view of Russell and McLean.

5). Whether Appellants' claims 15 and 17 are unpatentable under 35 U.S.C. § 103(a) over Zeanah in view of McLean.

6). Whether Appellants' claims 18-21 are unpatentable under 35 U.S.C. § 103(a) over Zeanah in view McMillan.

GROUPING OF CLAIMS

No groups of claims stand or fall together. Each of Appellants' claims recites at least one element, combination of elements, or step not found or suggested in the applied references, which patentably distinguishes the claims.

Every claim recites additional features of the invention which distinguishes the claim over every other pending claim.

The pending claims include seven independent claims (claims 1, 8, 9, 11, 12, 13, and 16). Claims 2-7 depend from claim 1. Claim 10 depends from claim 9. Claims 14-15 and 28-30 depend from claim 13. Claims 17-27 depend from claim 16. All pending claims are reproduced in the Appendix.

ARGUMENT

The Applicable Legal Standards

Anticipation pursuant to 35 U.S.C. § 102 requires that a single prior art reference contain all the elements of the claimed invention arranged in the manner recited in the claim. *Connell v. Sears, Roebuck & Co.*, 722 F.2d 1542, 1548, 220 USPQ 193, 198 (Fed. Cir. 1983).

Anticipation under 35 U.S.C. § 102 requires in a single prior art disclosure, each and every element of the claimed invention arranged in a manner such that the reference would literally infringe the claims at issue if made later in time. *Lewmar Marine, Inc. v. Barient, Inc.*, 822 F.2d 744, 747, 3 USPQ2d 1766, 1768 (Fed. Cir. 1987).

Before a claim may be rejected on the basis of obviousness pursuant to 35 U.S.C. § 103, the Patent Office bears the burden of establishing that all the recited features of the claim are known in the prior art. This is known as *prima facie* obviousness. To establish *prima facie* obviousness, it must be shown that all the elements and relationships recited in the claim are known in the prior art. If the Office does not produce a *prima facie* case, then the Appellants are under no obligation to submit evidence of nonobviousness. MPEP § 2142.

Even if all of the features recited in the claim are known in the prior art, it is still not proper to reject a claim on the basis of obviousness unless there is a specific teaching, suggestion, or motivation in the prior art to produce the claimed combination. *Panduit Corp. v. Denison Mfg. Co.*, 810 F.2d 1561, 1568, 1 USPQ2d 1593 (Fed. Cir. 1987). *In re Newell*, 891 F.2d 899, 901, 902, 13 USPQ2d 1248, 1250 (Fed. Cir. 1989).

The teaching, suggestion, or motivation to combine the features in prior art references must be clearly and particularly identified in such prior art to support a rejection on the basis of obviousness. It is not sufficient to offer a broad range of sources and make conclusory statements. *In re Dembiczak*, 50 USPQ2d 1614, 1617 (Fed. Cir. 1999).

It is respectfully submitted that the Action from which this appeal is taken does not meet these burdens.

The Zeanah Reference

Zeanah is directed to a system and method for delivering financial services. The disclosure of Zeanah (both the patent and the provisional application) is incomprehensible due to

lack of details concerning operation of the system. Due to Zeanah's lack of a disclosed operation, Appellants have been required to speculate as to how the Zeanah system could be made to operate. Therefore, the description of Zeanah herein or any comments directed thereto shall not be construed as agreement or an admission by Appellants that the Zeanah system is capable of operation or of achieving any of the functions carried out by Appellants' system.

Zeanah's arrangement appears to have a delivery system (12) intermediate plural remote devices (14, 16, 18, 20, 24) and other computer data systems (e.g., a bank's internal computer system). Zeanah's peripheral devices, such as a card reader (col. 9, lines 7-17) can be associated with a remote device (col. 7, lines 65-67). The remote devices are remote (distant) from the delivery system (12) (Figure 1). Zeanah requires that all of the remote devices (including ATMs) communicate directly through the delivery system (12) in order to provide services to the remote devices (Figure 1; col. 3, lines 63-67; col. 4, lines 54-56; col. 5, lines 44-60; col. 29, lines 20-35). In other words, all data to and from a remote device is through the delivery system (12). The remote devices are at the end of the line in Zeanah's services process. The remote devices apparently merely pass along input data to the delivery system (12) which then performs services on behalf of the remote devices. Zeanah's delivery system (12) is separate from the remote devices (col. 5, lines 44-60; Figure 1).

The Russell Reference

Russell is directed to a system for carrying out information transactions using Web documents. A URL (Uniform Resource Location) encoded bar code (8) is able to be read by a

bar code reader (7A). The bar code is decoded and the URL is used to access a document whose location is specified by the URL.

The McLean Reference

McLean is directed to a loader (12), cartridge (10), and a currency note dispenser (14). The loader is operative to load a predetermined number of notes into the cartridge. The cartridge is insertable into the dispenser. The cartridge is transferrable from the loader to the dispenser.

The McMillan Reference

McMillan is directed to an automated financial system. The system includes a central module (8) associated with plural ATM fascias (10, 12, 14). The ATM fascias are mounted on an exterior wall (16) of a building in which the central module is located. The central module includes the cash dispenser (40). Tracks (20) are used to transport currency from the central module to an ATM fascia.

(ii) 35 U.S.C. § 112, Second Paragraph

Claim 27 was rejected under 35 U.S.C. § 112, second paragraph, on the grounds of being indefinite. This rejection is respectfully traversed.

The Action acknowledges that claim 27 has a "step recited." Nevertheless, the Action alleges that the recited "step" of claim 27 is not "a complete operative method."

The Appellants respectfully disagree. A method claim "step" by itself is not required to be a "complete" method. Thus, the reason provided in the Action does not constitute a valid basis for a finding of indefiniteness. Further, one having ordinary skill in the art would recognize that claim 27 is not indefinite. In claim 27 the "carrying out" has antecedent basis in step (c) of claim 16. Hence, claim 27 recites that the "carrying out" is performed with at least one software applet.

Appellants respectfully submit that claim 27 is not indefinite. As shown herein, claim 27 meets the requirements of 35 U.S.C. § 112, second paragraph. Appellants respectfully submit that the 35 U.S.C. § 112, second paragraph, rejection is not appropriate and should be withdrawn.

Additionally, the 35 U.S.C. § 112, second paragraph, rejection of claim 27 is the only rejection of claim 27 presented in the Action. Therefore, since the only rejection against claim 27 has been shown to be inappropriate, Appellants respectfully submit that claim 27 is allowable.

(iii) 35 U.S.C. § 102

Appellants' remarks concerning the claim rejections in no way waive their rights to have the rejections relying on the Zeanah reference stricken for the reasons presented herein.

Zeanah Does Not Constitute Prior Art

Appellants have sworn behind the Zeanah reference

Zeanah apparently claims priority to provisional application 60/029,209 filed October 31, 1996. Appellants' claims patentably distinguish over the Zeanah reference. Nevertheless, the Declaration filed on February 4, 2002 pursuant to 37 C.F.R. § 1.131 swears behind the Zeanah reference. The Declaration swears back prior to July 7, 1996.

The Office entered the Declaration and acknowledged that it effectively swore behind both Downing (U.S. Patent 5,963,647) and Chang (U.S. Patent 5,884,288) as evidenced by the Office vacating the previous rejections based on these references (and the non use of these references in the Action). There is no contrary evidence of record. Appellants respectfully submit that the Declaration also effectively swears behind the Zeanah reference because Zeanah's earliest possible effective filing date is after July 7, 1996. Therefore, any rejection relying on the Zeanah reference has been rendered moot.

It follows that the Offices's reliance on the Zeanah reference is moot in view of the Declaration. That is, the relied upon Zeanah reference does not constitute prior art. Hence, it is respectfully submitted that the 35 U.S.C. § 102(e) (and 35 U.S.C. § 103(a)) rejections should be withdrawn.

Zeanah is not entitled to the provisional application filing date of October 31, 1996

As noted above, Zeanah does not constitute prior art because the Declaration swears behind the October 31, 1996 filing date of Zeanah's provisional application 60/029,209. Nevertheless, even if the Declaration were not filed, Zeanah would still not constitute prior art because Zeanah is not entitled to the filing date of the provisional application.

Zeanah's provisional application does not comply with the first paragraph of 35 U.S.C. § 112. Therefore, the Zeanah patent is only entitled to the nonprovisional application filing date of August 7, 1997. The present invention is entitled to (and claims benefit of) the November 27, 1996 filing date of provisional application 60/031,956. Thus, Zeanah does not constitute prior art against Appellants' invention.

Zeanah apparently claims priority to provisional application 60/029,209 filed October 31, 1996 (copy enclosed). However, a provisional application is required to comply with the first paragraph of 35 U.S.C. § 112. Note 35 U.S.C. § 111; 37 C.F.R. § 1.51; and MPEP § 601.

It is respectfully submitted that the provisional application of Zeanah does not meet the requirements of the first paragraph of 35 U.S.C. § 112. Zeanah's provisional application does not contain sufficient information to enable one skilled in the art on October 31, 1996 to make or use an operable system without undue experimentation. Thus, the specification of the Zeanah provisional application is not enabling. Therefore, the Zeanah patent is not entitled to the October 31, 1996 filing date of the provisional application, but rather the August 7, 1997 filing date of the nonprovisional application.

The First Paragraph of 35 U.S.C. § 112 Requirement

> "The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same, and shall set forth the best mode contemplated by the inventor of carrying out his invention."

The Undue Experimentation Factors

MPEP § 2164.01(a) lists factors to be considered when determining a lack of enablement requirement:

(A) The breadth of the claims.

(B) The nature of the invention.

(C) The state of the prior art.

(D) The level of one of ordinary skill in the art.

(E) The level of predictability in the art.

(F) The amount of direction provided by the inventor.

(G) The existence of working examples.

(H) The quantity of experimentation needed to make or use the invention based on the content of the disclosure.

Based on the evidence regarding each of the above factors, Zeanah's provisional application would not have taught one skilled in the art, at the date of filing, how to make and use the full scope of any invention without undue experimentation.

As can be seen, the Zeanah provisional application comprises a combination of three separate documents: first document (pages 1-9); second document (pages i to A-5); and third document (section numbered pages 1-47).

The first document (pages 1-9) is a requirements document of what the inventors would like to do, but lacks any presentation of knowledge of how to do it. This document merely provides proposed requirements for an alleged conceptual system. An example of the proposed

requirements may be found in section 2.2 (pages 4- 5) which states "1. A user should be"; "2. There should be a mechanism"; "3. The architecture should support"; "4. The architecture should support"; "5. The architecture should support"; and "6. The architecture should support".

Note that the Zeanah provisional application language does not state that "a user is able"; "there is a mechanism"; or that "the architecture does support". Also, the first document lacks any description of the specific details necessary for a reproducible operable embodiment. The first document does not appear to be of any inventive value.

Even if it were somehow possible for one skilled in the art to have ascertained the "requirements", the inventive step (which lies in reducing to practice a working relationship of structure to enable practice of the desired system, i.e., an enabling system) would still be lacking. Thus, the first document does not provide an enabling system.

The second document of the Zeanah provisional application (pages i to A-5) is an architecture section which lacks the technology and explanations corresponding therewith of how to practice the system. The second document itself admits (page 1-1) that "The primary focus of this document is the conceptual application architecture" and that "following the adaption of this conceptual architecture, the next level of design work will take place fleshing out the details." Section A.7 (pages A-4 and A-5) begins the "hypothetical examples that must assume some implementation details which will change when the detailed application framework design is done." That is, the provisional application reveals that after the alleged concept was completed (which it was not), then the next work (which hadn't even started yet) would involve figuring out the details (the guts of any inventive subject matter) necessary in order to implement the asserted

concept. Clearly one skilled in the art would not have been able to make and use the deficient incomplete concept (i.e., the alleged "invention") disclosed in the Zeanah provisional application without undue experimentation. Thus, the second document does not provide an enabling system.

The third document of the Zeanah provisional application (section numbered pages 1-47) comprises presentation overheads which again are evidence that the disclosure was merely a theoretical idea without any enablement or reduction to practice. Page 2 of the third section lists the five (5) process steps needed to reduce to practice the asserted conceptual idea. However, a close reading of the provisional application reveals that only step one (basic theoretical idea) was ever begun. Page 47 is evidence that steps two through four were yet to begin. Page 47, in concluding the provisional application, admits that the necessary "next steps" of obtaining an architecture, selecting the proper tools, defining an implementation strategy, staffing a design team, and starting the detailed design were not yet completed (or begun). That is, steps two through four were not completed (or even begun) as of the filing date of October 31, 1996. Also, it should be noted that step four (page 2) admits that the "how", which is an integral component of enablement, is lacking. Thus, the third document also does not provide an enabling system.

The three documents which make up the Zeanah provisional application, taken alone or in combination, fail to provide an enabling system. The provisional application's conceptual idea of "computer architecture to support global computer applications, including multiple languages and regional customizing" (title) is analogous to someone having the unresolved idea of a human travel to and from the nearest star. The provisional application's idea may have been noble, but

the necessary details for making the system were not known or included in the provisional filing on October 31, 1996.

"A conception of an invention, though evidenced by disclosure, drawings, and even a model, is not a complete invention under the patent laws" (MPEP § 715.07; page 700-211, col. 1, August 2001). As previously shown, the Zeanah provisional application is merely a conceptual idea document. Under U.S. patent laws it cannot constitute a complete invention. Thus, because it is incomplete, it cannot be enabling.

Furthermore, it is respectfully submitted that the conceptual idea as disclosed in the provisional application of Zeanah does not meet the patent law definition of "conception" (MPEP § 2138.04). Conception requires the complete performance of the mental part of the inventive act and it is the formation in the mind of the inventor of a definite and permanent idea of the complete and operative invention as it is thereafter to be applied in practice. Conception requires the disclosure to be clear to enable one skilled in the art to reduce it to practice without undue experimentation or inventive skill. Conception is not complete if uncertainty undermines the specificity of the inventor's idea that it is not yet a definite and permanent reflection of the complete invention as it will be used in practice. As previously discussed, Zeanah's provisional application is neither complete nor operative, and it would require undue experimentation and further inventive skill. Thus, Zeanah's provisional application disclosure cannot constitute "conception."

Appellants have shown that Zeanah's provisional application does not constitute a "conception." Furthermore, even if it were somehow possible for the provisional application to

meet the requirements of "conception", "conception is not enablement" (MPEP § 2138.04; page 2100-107, col. 1, August 2001). Nevertheless, Appellants have also shown that the provisional application does not meet the "enabling" requirements of the first paragraph of 35 U.S.C. § 112.

Therefore, the Zeanah patent is not entitled to the October 31, 1996 filing date of the non enabling provisional application. The Zeanah patent is, at best, only entitled to the August 7, 1997 filing date. The application that is the subject of this appeal is entitled to (and claims the benefit of) the November 27, 1996 filing date of provisional application 60/031,956. In conclusion, the Zeanah reference cannot constitute prior art against Appellants' invention.

The Pending Claims Are Not Anticipated By Zeanah

In the Action claims 1-6, 8-13, 16, and 22-26 were rejected under 35 U.S.C. § 102(e) as being anticipated by Zeanah. These rejections are respectfully traversed.

As noted above, Zeanah does not constitute prior art. Nevertheless, even if it were somehow possible for Zeanah to constitute prior art, Zeanah still would not anticipate the claimed invention.

Zeanah does not teach each and every feature, relationship and/or step of the claimed invention arranged in the manner recited in the claims, as is required to sustain the rejections. Thus, it is respectfully submitted that the 35 U.S.C. § 102(e) rejections should be withdrawn.

Claim 1

Zeanah does not teach an automated banking machine including a transaction function device and a computer and software in the manner recited. Zeanah does not teach software that

is operative to process HTML documents including instructions therein in the manner recited. Zeanah does not teach that a transaction function device is operative to carry out a transaction function responsive to software processing at least one document including at least one instruction adapted to cause a computer to cause operation of the transaction function device. Zeanah does not teach processing an HTML document instruction to cause operation of a transaction function device.

The Action (on page 3) relies on Zeanah's reference to a web browser (col. 5, lines 50-53). However, there is no evidence in Zeanah of an HTML document including an instruction to cause operation of a transaction function device. Nor is there any evidence in Zeanah that a browser processes HTML document instructions to operate a transaction function device. The Action alleges that "web browsers are software application used to locate and display web pages; hence, 'processing HTML documents.'" The Action further alleges that "the transaction function device is operative to carry out the transaction function responsive to the browser processing at least one document including at least one instruction adapted to cause the computer to cause operation of the transaction function device."

The Appellants respectfully disagree. Claim 1 specifically recites that "the transaction function device is selectively operative to carry out a transaction function." Claim 1 also specifically recites that "the input device is operative to receive inputs, whereby a user is enabled to provide inputs to the machine" and that "the output device outputs information, whereby a user is enabled to perceive outputs from the output device." The Action is silent as to what components of Zeanah constitute the recited input device and output device. However, the

Action does allege that Zeanah at col. 9, lines 8-14 teaches a transaction function device. Appellants respectfully submit that the referenced "touch screen" and "screen display" (at col. 9, lines 8-14) do not constitute transaction function devices in Zeanah, but (at best) respectively an input device and an output device. The other referenced components (printer, card reader, envelope depository, cash dispenser, etc.) do not involve display. Thus, even if the Action's allegation that "web browsers are software application used to locate and display web pages" was accurate, then Zeanah's web browser would not be associated with non display components (printer, card reader, envelope depository, cash dispenser, etc.). That is, Zeanah's non display components do not require or use the display of web pages (i.e., the use of Zeanah's browser) to operate. For example, why would a cash dispenser need a browser to display a web page (as alleged in the Action) in order to operate? Hence, Zeanah's browser is unrelated to operation of a transaction function device. Further, there is no evidence that Zeanah's browser can process transaction function device operating instructions. Zeanah's browser, at best, and as admitted in the Action, is only operative to "display web pages." That is, Zeanah's browser does not include software which is capable of performing transaction function device instruction processing. It follows that Zeanah does not teach the recited software. It further follows that Zeanah does not teach that a transaction function device operates responsive to a browser processing an HTML document instruction.

The Action's allegation that "the transaction function device is operative to carry out the transaction function responsive to the browser processing at least one document including at least one instruction adapted to cause the computer to cause operation of the transaction function

device (see col. 5, lines 44-54; col. 9, lines 8-14; col. 14, lines 49-54; col. 32, lines 48-57)" is without merit. None of these cited sections of Zeanah teach the allegation of a browser processing a document instruction to operate a transaction function device. As previously discussed, Zeanah's browser (at best) is for displaying web pages, it is not associated with processing transaction function device operating instructions. That is, Zeanah does not teach the recited processing software.

The Action's references to Zeanah at col. 5 and col. 9 have already been discussed. Zeanah at col. 14, lines 49-54 and col. 32, lines 48-57 pertains to the transaction services set (90). A transaction executor component (91) validates data to determine whether enough information actually exists for a transaction. If data is missing then additional information is collected. Information may be collected from other objects, such as a customer ID component (111). The transaction services set (90) is not associated with software which is capable of performing transaction function device instruction processing. Nor can the transaction services set (90) constitute the recited software. Contrarily, the transaction services set (90) is located in Zeanah's delivery system (12) (e.g., Figure 2) which is separate (distant) from the remote devices (e.g., Figure 1) (in which the Action alleges the software is contained). That is, the transaction services set (90) would not be located at an automated banking machine, especially an automated banking machine including a transaction function device and software operative to process HTML document instructions to cause operation of the transaction function device.

Appellants respectfully submit that Zeanah does not disclose each and every element of the claimed invention arranged in the manner recited in the claim, as is required to sustain the

rejection. The rejection of claim 1 is based on alleged teachings of Zeanah, not factual showings of what Zeanah actually teaches. Therefore, Zeanah cannot anticipate claim 1.

Claim 2

Zeanah does not teach that a sheet dispenser is operative to function responsive to a browser processing a document instruction adapted to cause operation of the sheet dispenser. Zeanah does not anticipate the claim.

Claim 3

Zeanah does not teach that a card reader is operative to function responsive to a browser processing a document instruction adapted to cause operation of the card reader. Zeanah does not anticipate the claim.

Claim 4

Zeanah does not teach that a printer is operative to function responsive to a browser processing a document instruction adapted to cause operation of the printer. Zeanah does not anticipate the claim.

Claim 5

Zeanah does not teach that a depository is operative to function responsive to a browser processing a document instruction adapted to cause operation of the depository. Zeanah does not anticipate the claim.

Claim 6

Zeanah does not teach that a keyboard is operative to function responsive to a browser processing a document instruction adapted to cause operation of the keyboard. Zeanah does not anticipate the claim.

Claim 8

Claim 8 is directed to an ATM. The Action's allegations and the relied upon sections concerning Zeanah have already been addressed with regard to claim 1. Thus, Appellants' remarks in support of the patentability of claim 1 are incorporated by reference as if fully rewritten herein. Zeanah does not teach that a transaction function device is operative to cause the ATM to carry out a transaction function responsive to at least one HTML document that is received by a browser. Hence, Zeanah does not anticipate claim 8.

Claim 9

Appellants' remarks in support of the patentability of claim 1 are incorporated by reference as if fully rewritten herein.

Claim 9 is directed to a method. Zeanah does not teach an HTML format document including a transaction instruction. Zeanah also does not teach receiving the HTML format document with a browser. Zeanah further does not teach carrying out a transaction function with a transaction function device in an automated banking machine responsive to the transaction instruction of the document. As previously discussed, Zeanah's browser, at best, and as admitted in the Action, is only operative to display web pages, it is not associated with receiving transaction instructions. Thus, Zeanah cannot anticipate claim 9.

Claim 10

Claim 10 depends from claim 9. Zeanah does not also produce an output through an output device (of an automated banking machine) responsive to the at least one HTML document. That is, Zeanah does not both carry out a transaction function with a transaction function device and produce an output through an output device responsive to at least one HTML document. Thus, Zeanah does not anticipate the claim.

Claim 11

Appellants' remarks in support of the patentability of claim 1 are incorporated by reference as if fully rewritten herein.

Claim 11 is directed to a method. Zeanah does not teach carrying out a transaction function with a transaction function device in an automated banking machine responsive to a document having a transaction instruction embedded therein. Where does Zeanah teach the recited step? Zeanah does not anticipate claim 11.

Claim 12

Appellants' remarks in support of the patentability of claim 1 are incorporated by reference as if fully rewritten herein.

Claim 12 is directed to an ATM. Zeanah does not teach an ATM that operates to conduct a financial transaction responsive to a mark-up language document. Therefore, Zeanah cannot anticipate claim 12.

Claim 13

Appellants' remarks in support of the patentability of claim 1 are incorporated by reference as if fully rewritten herein.

Claim 13 is directed to an automated banking machine. Zeanah does not teach that a computer is adapted to cause a banking transaction to be carried out through operation of at least one transaction function device in the machine responsive to at least one mark up language document. Consequently, Zeanah cannot anticipate claim 13.

Claim 16

Appellants' remarks in support of the patentability of claim 1 are incorporated by reference as if fully rewritten herein.

Claim 16 is directed to a method. Zeanah does not teach carrying out at least a portion of a banking transaction with an automated banking machine transaction function device responsive to processing at least one mark up language document with a computer. Zeanah does not teach the recited steps. It follows that Zeanah cannot anticipate claim 16.

Claim 22

Claim 22 depends from claim 16. Zeanah does not provide an output through an output device (of an automated banking machine) responsive to processing at least one mark up language document with the computer in the manner recited. That is, Zeanah does not both carry out at least a portion of a banking transaction with a transaction function device and provide an output through an output device responsive to processing at least one mark up language document with a computer. Thus, Zeanah does not anticipate the claim.

Claim 23

Claim 23 depends from claim 22. Zeanah does not teach that an output is provided responsive to browser software processing the at least one mark up language document. Zeanah does not anticipate the claim.

Claim 24

Claim 24 depends from claim 23. Zeanah does not teach providing visual output through a screen responsive to browser software processing the at least one mark up language document. Zeanah does not anticipate the claim.

Claim 25

Claim 25 depends from claim 16. Zeanah does not teach that an HTML document is processed by a computer in the manner recited. Thus, Zeanah does not anticipate the claim.

Claim 26

Claim 26 depends from claim 16. Zeanah does not teach that the processing of at least one mark up language document is operative to both provide an output through an output device and to carry out at least a portion of a banking transaction. It follows that Zeanah cannot anticipate the claim.

(iv) 35 U.S.C. § 103

The attempts to combine the teachings of the references are clearly attempts at hindsight reconstruction of Appellants' claimed invention, which is legally impermissible and does not constitute a valid basis for a finding of obviousness. *In re Fritch*, 22 USPQ2d 1780 (Fed. Cir.

1992). The rejections, which lack the necessary evidence and rationale, are based on knowledge gleaned only from Appellants' disclosure. It follows that it would not have been obvious to have modified the references in the manner alleged. Furthermore, without a motivation to combine, which is the current situation, a rejection based on a *prima facie* case of obviousness is improper (MPEP § 2143.01).

Appellants traverse the rejections on the grounds that Appellants' claims recite features, relationships, and/or steps which are neither disclosed nor suggested in the prior art, and because there is no teaching, suggestion, or motivation cited so as to produce Appellants' invention. The features, relationships, and/or steps recited in Appellants' claims patentably distinguish over the applied reference(s). Nor would it have been obvious to one having ordinary skill in the art to have combined the teachings of the references to have produced the recited invention. The Office does not factually support any *prima facie* conclusion of obviousness. To establish *prima facie* obviousness, the prior art must teach or suggest all the claim limitations. If the Office does not produce a *prima facie* case, which is the current situation, then the Appellants are under no obligation to submit evidence of nonobviousness (MPEP § 2142). Thus, it is respectfully submitted that the 35 U.S.C. § 103(a) rejections should be withdrawn.

The legality of Russell as prior art is questioned

The Russell patent has a filing date of August 22, 1997. As previously discussed, Appellants' Declaration swears back prior to July 7, 1996. The Russell patent is a continuation-in-part of many other applications, some of which themselves are a continuation-in-part of

another application. Some of these applications dates are not earlier than July 7, 1996. For example, the Russell patent is a continuation-in-part of application 08/838,501 (filed April 7, 1997) which is a continuation-in-part of application 08/820,540 (filed March 19, 1997) which is a continuation-in-part of application 08/753,367 (filed November 25, 1996). The Russell patent indicates that applications 08/838,501 and 08/820,540 relate to bar codes and the Internet (col. 1). Hence, Appellants have shown the likelihood that the subject matter relied upon in the Russell patent is not prior art. The Office has presented no evidence of record that the subject matter relied upon in the Russell patent is prior art. Appellants are entitled to a showing of facts. Thus, Appellants respectfully submit that the Russell patent does not constitute prior art.

The Pending Claims Are Not Obvious Over
Zeanah in view of Russell

In the Action claims 7, 14, 28, and 30 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Zeanah in view of Russell. These rejections are respectfully traversed.

The independent claims from which these claims depend have been previously shown to be allowable. Thus, it is asserted that these dependent claims are allowable on the same basis. Furthermore, each of these dependent claims additionally recites specific features and relationships that patentably distinguish the claimed invention over the applied art.

Neither Zeanah nor Russell, taken alone or in combination, discloses or suggests the recited features and relationships. Nor would it have been obvious to one having ordinary skill in the art to have modified Zeanah with the teachings of Russell to have produced the claimed

invention. The Action does not establish a *prima facie* showing of obviousness. Therefore, Appellants respectfully submit that the 35 U.S.C. § 103(a) rejections should be withdrawn.

Claim 7

Claim 7 depends from claim 1. The Action admits (on page 8) that Zeanah does "not expressly disclose a software that is operative responsive to an instruction to access at least one HTTP record address (i.e. "URL"), wherein the at least one HTTP record address corresponds to at least one HTTP record (i.e. "HTML document") including instructions adapted to cause the computer to cause operation of the transaction function device."

The Action alleges that Russell teaches the features and relationships admitted as absent in Zeanah. Appellants respectfully disagree. Russell cannot overcome both the admitted and the other (previously discussed) deficiencies of Zeanah as it does not disclose or suggest the recited features and relationships which are not found in Zeanah.

The Action is unclear as to where Russell teaches or suggests accessing an HTTP record having instructions adapted to cause operation of a transaction function device, especially where software is operative responsive to an instruction to access an HTTP record address corresponding to the HTTP record. The cited sections of Russell do not teach or suggest the features and relationships. Nor does Russell teach or suggest the features and relationships. For example, where does Russell teach or suggest the relationship of software and an HTTP record having instructions adapted to cause operation of a transaction function device?

The Action is silent as to how the system of Zeanah could be modified by the teachings of Russell. Nor is there any motivation to combine. Also, the alleged modification to Zeanah

would destroy the disclosed and desired utility and operability of the Zeanah teaching. An obviousness rejection cannot be based on a combination of features if making the combination would result in destroying the utility or advantage of the device shown in the prior art. *In re Fine,* 5 USPQ2d 1598-99 (Fed. Cir. 1988). The only suggestion for having the recited software is found in Appellants' own novel disclosure. It follows that the alleged modification of Zeanah (and the rejection) is based on hindsight reconstruction of the recited invention based on Appellants' disclosure, which is impermissible. Furthermore, even if it were somehow possible to modify the system of Zeanah with the teachings of Russell as alleged, such modification still would not result in the recited features and relationships. Neither Zeanah nor Russell, taken alone or in combination, discloses or suggests the recited features and relationships. Therefore, it would not have been obvious to one having ordinary skill in the art to have modified Zeanah with the teachings of Russell to have produced the claimed invention. The Action has not presented a *prima facie* showing of obviousness.

Claim 14

Claim 14 depends from claim 13. The Action admits (on page 9) that Zeanah does "not disclose a computer including a document handling software." Appellants respectfully submit that Russell cannot overcome both the admitted and previously discussed deficiencies of Zeanah, as it does not disclose or suggest the recited features and relationships which are not found in Zeanah. For example, Russell does not disclose or suggest that a computer is operative to carry out a banking transaction responsive to document handling software processing a mark up language document, especially where the computer is adapted to cause the banking transaction to

be carried out through operation of a transaction function device responsive to the mark up language document. Neither Zeanah nor Russell, taken alone or in combination, discloses or suggests the recited features and relationships. Nor has the Action presented a *prima facie* showing of obviousness.

Claim 28

Claim 28 depends from claim 14. As previously discussed with regard to claim 14, the Action admits (on page 9) that Zeanah does "not disclose a computer including a document handling software." As previously discussed, it would not have been obvious to one having ordinary skill in the art to have modified Zeanah with the teachings of Russell to have produced the recited invention of claim 14.

Additionally, neither Zeanah nor Russell, taken alone or in combination, discloses or suggests that a computer is adapted to automatically operate a transaction function device responsive to the processing of a mark up language document with document handling software. The relied upon sections of Russell at col. 6, lines 39-46 and col. 21, lines 43-45 do not teach or suggest automatically operating a transaction function device responsive to the processing of a mark up language document. Thus, it would not have been obvious to one having ordinary skill in the art to have modified Zeanah with the teachings of Russell to have produced the claimed invention. It follows that the Action has not established a *prima facie* showing of obviousness.

Claim 30

Claim 30 depends from claim 13. Neither Zeanah nor Russell, taken alone or in combination, discloses or suggests that a computer is operative both to cause a banking

transaction to be carried out through operation of a transaction function device responsive to at least one mark up language document, and to automatically display at least one visual output through a display device responsive to processing of the at least one mark up language document. The relied upon sections of Russell do not teach or suggest recited features and relationships. The Action is silent as to how the system of Zeanah could be modified by the teachings of Russell. Nor would it have been obvious to have modified Zeanah with the teachings of Russell to have produced the claimed invention. Again, the Office has not presented a *prima facie* showing of obviousness.

The Pending Claims Are Not Obvious Over
Zeanah in view of Russell and McLean

In the Action claim 29 was rejected under 35 U.S.C. § 103(a) as being unpatentable over Zeanah in view of Russell and McLean. This rejection is respectfully traversed.

Claim 29

Claim 29 depends from claim 28 which depends from claim 14 which depends from claim 13. The Action admits (on page 10) that Zeanah does not disclose a currency dispenser dispensing at least one note. That is, the Action admits that Zeanah does not teach the dispensing of currency through operation of a currency dispenser responsive to at least one mark up language document.

Neither Russell nor McLean teaches or suggests the dispensing of currency through operation of a currency dispenser responsive to at least one mark up language document. The

relied upon sections of McLean do not teach or suggest the recited features and relationships. The Action is silent as to McLean even being associated with a mark up language document. Thus, neither Russell nor McLean can overcome both the admitted and previously discussed deficiencies of Zeanah as neither reference discloses nor suggests the recited features and relationships which are not found in Zeanah.

Neither Zeanah nor Russell nor McLean, taken alone or in combination, teaches or suggests the recited features and relationships. Thus, it would not have been obvious to have modified Zeanah with the teachings of Russell and McLean to have produced the claimed invention. Nor has the Office presented a *prima facie* showing of obviousness.

The Pending Claims Are Not Obvious Over
Zeanah in view of McLean

In the Action claims 15 and 17 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Zeanah in view of McLean. These rejections are respectfully traversed.

Claim 15

Claim 15 depends from claim 13. The Action admits (on page 10) that Zeanah does not disclose dispensing at least one note. That is, the Action admits that Zeanah does not teach the dispensing of a note through operation of a note dispenser responsive to at least one mark up language document.

McLean does not teach or suggest the dispensing of a note through operation of a note dispenser responsive to at least one mark up language document. The relied upon sections of

McLean do not teach or suggest the recited features and relationships. The Action is silent as to McLean even being associated with a mark up language document. Thus, McLean cannot overcome both the admitted and previously discussed deficiencies of Zeanah, as it does not disclose or suggest the recited features and relationships which are not found in Zeanah.

Neither Zeanah nor McLean, taken alone or in combination, teaches or suggests the recited features and relationships. Thus, it would not have been obvious to have modified Zeanah with the teachings of McLean to have produced the claimed invention. The Office has not established a *prima facie* showing of obviousness.

Claim 17

Claim 17 depends from claim 16. The Action admits (on page 11) that Zeanah does not disclose dispensing at least one note. That is, the Action admits that Zeanah does not teach dispensing a note with a note dispenser responsive to processing a mark up language document.

McLean does not teach or suggest the dispensing of a note with a note dispenser responsive to processing a mark up language document. The relied upon sections of McLean do not teach or suggest the recited features, relationships, and steps. The Action is silent as to McLean even being associated with a mark up language document. Thus, McLean cannot overcome both the admitted and previously discussed deficiencies of Zeanah, as it does not disclose or suggest the recited features and relationships which are not found in Zeanah.

Neither Zeanah nor McLean, taken alone or in combination, teaches or suggests the recited features and relationships. It follows that it would not have been obvious to have

modified Zeanah with the teachings of McLean to have produced the claimed invention. Thus, the Office has not established a *prima facie* showing of obviousness.

The Pending Claims Are Not Obvious Over
Zeanah in view of McMillan

In the Action claims 18-21 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Zeanah in view of McMillan. These rejections are respectfully traversed.

Claim 18

Claim 18 depends from claim 16. The Action admits (on page 11) that Zeanah does not disclose a card reader reading indicia from a card. That is, the Action admits that Zeanah does not teach reading indicia with a reading device responsive to processing a mark up language document.

McMillan does not teach or suggest reading indicia with a reading device responsive to processing a mark up language document. The relied upon sections of McMillan do not teach or suggest the recited features, relationships, and steps. The Action is silent as to McMillan even being associated with a mark up language document. Thus, McMillan cannot overcome both the admitted and previously discussed deficiencies of Zeanah, as it does not disclose or suggest the recited features and relationships which are not found in Zeanah.

Neither Zeanah nor McMillan, taken alone or in combination, teaches or suggests the recited features and relationships. Nor would it have been obvious to have modified Zeanah with

the teachings of McMillan to have produced the claimed invention. The Office has not established a *prima facie* showing of obviousness.

Claim 19

Claim 19 depends from claim 18 which depends from claim 16. As previously discussed, the Action admits (on page 11) that Zeanah does not disclose a card reader reading indicia from a card. That is, the Action admits that Zeanah does not teach reading indicia with a card reader responsive to processing a mark up language document.

McMillan does not teach or suggest reading indicia with a card reader responsive to processing a mark up language document. The relied upon sections of McMillan do not teach or suggest the recited features, relationships, and steps. The Action is silent as to McMillan even being associated with a mark up language document. Thus, McMillan cannot overcome both the admitted and previously discussed deficiencies of Zeanah, as it does not disclose or suggest the recited features and relationships which are not found in Zeanah.

Neither Zeanah nor McMillan, taken alone or in combination, teaches or suggests the recited features and relationships. It follows that it would not have been obvious to have modified Zeanah with the teachings of McMillan to have produced the claimed invention. Thus, the Office has not established a *prima facie* showing of obviousness.

Claim 20

Claim 20 depends from claim 16. Neither Zeanah nor McMillan, taken alone or in combination, teaches or suggests the recited features and relationships. Neither Zeanah nor McMillan, taken alone or in combination, teaches or suggests sensing a key input responsive to

processing a mark up language document. The Action is silent as to McMillan even being associated with a mark up language document. Thus, it would not have been obvious to have modified Zeanah with the teachings of McMillan to have produced the claimed invention. Nor has the Office presented a *prima facie* showing of obviousness.

Claim 21

Claim 21 depends from claim 16. Neither Zeanah nor McMillan, taken alone or in combination, teaches or suggests the recited features and relationships. Neither Zeanah nor McMillan, taken alone or in combination, teaches or suggests receiving a deposit with a depository responsive to processing a mark up language document. The Action is silent as to McMillan even being associated with a mark up language document. Thus, it would not have been obvious to have modified Zeanah with the teachings of McMillan to have produced the claimed invention. Again, the Office has not presented a *prima facie* showing of obviousness.

Additional Comments

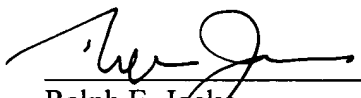
Appellants respectfully submit that the Action does not correctly ascertain the level of one having ordinary skill in the art, to which the claimed subject matter pertains, at the time of the invention. That is, the level of one having ordinary skill in the art is at least limited to that knowledge which was publically known prior to July 7, 1996 (note Appellants' Declaration). Thus, the Action's alleged motivations for combining reference teachings (and other allegations), which depend on one having ordinary skill in the art, lack the required (prior art) showing of evidence and rationale necessary to maintain the rejections. For example, the Action alleges (in

apparently all of the prior art rejections) that "it is known" that web browsers are software application used to locate and display web pages (i.e. 'HTML documents'). However, there is no indication that the Office stepped backward in time to when the invention was unknown and just before it was made to make its determination of reasons for motivation (and obviousness). Appellants respectfully submit that the Office did not reach a conclusion based on facts gleaned only from the prior art. That is, the Office incorrectly applied the level of one having ordinary skill in the art of today, especially in relation to use of HTML documents associated with the operation of a transaction function device in an automated banking machine. For these additional reasons, Appellants respectfully submit that the rejections should be withdrawn.

CONCLUSION

Each of Appellants' claims meets the requirements of 35 U.S.C. § 112. Also, as explained above, each of the pending claims specifically recites features, relationships, and/or steps that are neither disclosed nor suggested in any of the applied art. Furthermore, the applied art is devoid of any teaching, suggestion, or motivation for combining features of the applied art so as to produce the recited invention. For these reasons it is respectfully submitted that all of the pending claims are allowable.

Respectfully submitted,



Ralph E. Jocke Reg. No. 31,029
WALKER & JOCKE
231 South Broadway
Medina, Ohio 44256
(330) 721-0000

APPENDIX

CLAIMS

1. Apparatus comprising:

an automated banking machine, including:

an output device, wherein the output device outputs information, whereby a user is enabled to perceive outputs from the output device;

an input device, wherein the input device is operative to receive inputs, whereby a user is enabled to provide inputs to the machine;

a transaction function device, wherein the transaction function device is selectively operative to carry out a transaction function;

a computer, wherein the computer is in operative connection with the output device, the input device and the transaction function device;

software executable in the computer, wherein the software includes a browser, wherein the browser is operative to process HTML documents including instructions therein, and wherein the transaction function device is operative to carry out the transaction function responsive to the browser processing at least one document including at least one instruction adapted to cause the computer to cause operation of the transaction function device.

2. The apparatus according to claim 1 wherein the transaction function device includes a sheet dispenser.

3. The apparatus according to claim 1 wherein the transaction function device includes a card reader.

4. The apparatus according to claim 1 wherein the transaction function device includes a printer.

5. The apparatus according to claim 1 wherein the transaction function device includes a depository.

6. The apparatus according to claim 1 wherein the transaction function device includes a keyboard.

7. The apparatus according to claim 1 wherein the software is operative responsive to an instruction to access at least one HTTP record address, wherein the at least one HTTP record address corresponds to at least one HTTP record including instructions adapted to cause the computer to cause operation of the transaction function device.

8. An Automated Teller Machine (ATM) comprising:

a computer;

a browser operating in the computer;

a transaction function device in operative connection with the computer, wherein the transaction function device is operative to cause the ATM to carry out a transaction function responsive to at least one HTML format document that is received by the browser.

9. A method comprising the steps of:

a) operating a browser in at least one computer in connection with an automated banking machine;

b) receiving at least one HTML format document with the browser, wherein the at least one HTML format document includes at least one transaction instruction;

c) carrying out at least one transaction function with a transaction function device in the automated banking machine responsive to the at least one HTML format document.

10. The method according to claim 9 wherein the automated banking machine includes an output device in operative connection with the computer, and further comprising the step of:

d) producing an output through the output device responsive to the at least one HTML format document.

11. A method comprising the steps of:

a) operating a browser in at least one computer in operative connection with an automated banking machine;

b) receiving at least one document with the browser, wherein the document includes at least one transaction instruction embedded therein;

c) carrying out at least one transaction function with a transaction function device in the automated banking machine responsive to the at least one document including the at least one transaction instruction.

12. An Automated Teller Machine (ATM) that operates to conduct at least one financial transaction responsive to at least one mark-up language document.

13. An automated banking machine comprising:

a computer in operative connection with the banking machine;

at least one transaction function device in the banking machine adapted to carry out at least a portion of a banking transaction;

wherein the computer is adapted to cause at least one banking transaction to be carried out through operation of the at least one transaction function device responsive to at least one mark up language document.

14. The machine according to claim 13 wherein the computer includes document handling software, and wherein the computer is operative to carry out the at least one banking transaction responsive to the document handling software processing the at least one mark up language document.

15. The automated banking machine according to claim 13 wherein the transaction function device includes a note dispenser, and wherein the at least one banking transaction includes dispensing at least one note from the note dispenser.

16. A method comprising the steps of:

a) providing an automated banking machine including at least one transaction function device, wherein the automated banking machine is in operative connection with at least one computer;

b) processing at least one mark up language document with the computer;

c) carrying out at least a portion of a banking transaction with the transaction function device responsive to processing the at least one mark up language document with the computer in step (b).

17. The method according to claim 16 wherein the transaction function device includes a note dispenser, and wherein in step (c) the portion of the banking transaction includes dispensing at least one note with the note dispenser.

18. The method according to claim 16 wherein the transaction function device includes at least one reader device, and wherein in step (c) the portion of the transaction includes reading indicia with the reading device.

19. The method according to claim 18 wherein the reading device includes a card reader, and wherein in step (c) indicia is read from a card.

20. The method according to claim 16 wherein the transaction function device includes at least one key, and wherein in step (c) the portion of the banking transaction includes sensing an input through the at least one key.

21. The method according to claim 16 wherein the transaction function device includes a depository, and wherein in step (c) the portion of the banking transaction includes receiving a deposit with the depository.

22. The method according to claim 16 wherein in step (a) the banking machine includes at least one output device, and further comprising the step of:

d) providing at least one output through the output device responsive to processing at least one mark up language document with the computer.

23. The method according to claim 22 wherein the computer includes browser software, and wherein in step (d) the at least one output is provided responsive to the browser software processing the at least one mark up language document.

24. The method according to claim 23 wherein the output device includes a screen and wherein in step (d) the at least one output includes a visual output through the screen.

25. The method according to claim 16 wherein in step (b) at least one HTML document is processed by the computer.

26. The method according to claim 16 wherein the automated banking machine includes an output device, and wherein in step (c) processing the at least one mark up language document is operative to cause the computer to provide an output through the output device and to carry out at least the portion of the banking transaction.

27. The method according to claim 16 wherein the computer is operative to cause the carrying out of the portion of the banking transaction responsive to at least one software applet.

28. The apparatus according to claim 14 wherein the document handling software includes a browser, wherein the computer is adapted to automatically operate at least one transaction function device responsive to the processing of at least one mark up language document with the document handling software.

29. The apparatus according to claim 28 wherein at least one transaction function device includes a currency sheet dispenser, and wherein the banking transaction includes dispensing at least one currency sheet from the currency sheet dispenser.

30. The apparatus according to claim 13 and further including a display device having a display screen, wherein the computer includes document handling software, wherein the computer is operative to automatically display at least one visual output through the display device responsive to processing at least one mark up language document with the computer.

60/029209	Class	Subclass	ISSUE CLASSIFICATION

[redacted]

PROVISIONAL APPLICATION NUMBER
60/029209

SERIAL NUMBER	FILING DATE	CLASS	SUBCLASS	GROUP ART UNIT	EXAMINER
60/029,209 PROVISIONAL	10/31/96				

APPLICANTS

JAMES ZEANAH, THOUSAND OAKS, CA; CHARLES ABBOTT, SANTA MONICA, CA; NIK BOYD, LOS ANGELES, CA; ALBERT COHEN, LOS ANGELES, CA; JAMES COOK, MANHATTAN BEACH, CA; MICHAEL GRANDCOLAS, SANTA MONICA, CA; LAN SIKHUN, LOS ANGELES, CA; BONNIE L. LINDSLEY, SANTA CLARITA, CA; GRIGOR MARKARIAN, AGOURA, CA; LESLIE MOSS, LOS ANGELES, CA.

CONTINUING DATA********
VERIFIED

_____

FOREIGN/PCT APPLICATIONS********
VERIFIED

_____

FOREIGN FILING LICENSE GRANTED 11/27/96

Foreign priority claimed ☐ yes ☐ no 35 USC 119 conditions met ☐ yes ☐ no Verified and Acknowledged _____ Examiner's Initials	AS FILED →	STATE OR COUNTRY	SHEETS DRWGS.	TOTAL CLAIMS	INDEP. CLAIMS	FILING FEE RECEIVED	ATTORNEY'S DOCKET NO.
		CA	46			$150.00	T0091-097776

ADDRESS

DALE CURTIS HOGUE SR
KILPATRICK & CODY
SUITE 800
700 13TH STREET NW
WASHINGTON DC 20005

TITLE

COMPUTER ARCHITECTURE TO SUPPORT GLOBAL COMPUTER APPLICATIONS, INCLUDING MULTIPLE LANGUAGES AND REGIONAL CUSTOMIZING

U.S. DEPT. OF COMM./ PAT. & TM—PTO-436L (Rev.12-94)

Form PTO-1625
(Rev. 5/95

(FACE)

PATENT APPLICATION

60029209

APPROVED FOR LICENSE ☐

INITIALS NOV 1 9 2001

CONTENTS

(FRONT)

POSITION		ID NO.	DATE
CLASSIFIER		5	11.19.96
EXAMINER	459		
TYPIST		484	11-27
VERIFIER		15	11-29
CORPS CORR.			
SPEC. HAND			
FILE MAINT			
DRAFTING			

(LEFT INSIDE)

PATENT APPLICATION SERIAL NO. 60/029209

U.S. DEPARTMENT OF COMMERCE
PATENT AND TRADEMARK OFFICE
FEE RECORD SHEET

PTO-1556
(5/87)

#46

150-114

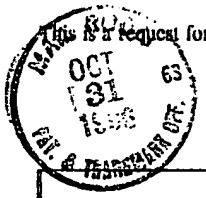
Patent

PTO/SB/16 (6-95)
Approved for use through 04/11/98. OMB 0651-0037
Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE

PROVISIONAL APPLICATION COVER SHEET

60/029209

This is a request for filing a PROVISIONAL APPLICATION under 37 CFR 1.53(b)(2).



Docket Number	T0091-097776	Type a plus sign (+) inside this box	+

INVENTOR(s)/APPLICANT(s)

LAST NAME	FIRST NAME	MIDDLE INITIAL	RESIDENCE (CITY AND EITHER STATE OR FOREIGN COUNTRY)
Zeanah	James		Thousand Oaks, California
	SEE ATTACHED PAGE FOR ADDITIONAL INVENTORS		

TITLE OF THE INVENTION (280 characters max)

Computer Architecture to Support Global Computer Applications, Including Multiple Languages and Regional Customizing

CORRESPONDENCE ADDRESS

Dale Curtis Hogue, Sr., Kilpatrick & Cody, L.L.P., 700 13th Street, N.W., Suite 800, Washington

State	D.C.	Zip Code	20005	Country	U.S.

ENCLOSED APPLICATION PARTS (check all that apply)

X	Specification	Number of Pages	75			Small Entity Statement
X	Drawing(s)	Number of Sheets	46			Other (specify)

METHOD OF PAYMENT (check one)

X	A check or money order is enclosed to cover the Provisional filing fees	Provisional Filing Fee Amount ($)	$150.00
	The Commissioner is hereby authorized to charge filing fees and credit Deposit Account Number: 11-0855		

The invention was made by an agency of the United States Government or under a contract with an agency of the United States Government.

[X] No

[ ] Yes, the name of the U.S. Government agency and the Government contract number are: _____

Respectfully submitted,

SIGNATURE: _____ 33,014

Date: 10/31/96

TYPED or PRINTED NAME: Dale Curtis Hogue, Sr.

REGISTRATION NO. 32,823 (if appropriate)

[X] Additional inventors are being named on separately numbered sheets attached hereto

PROVISIONAL APPLICATION FILING ONLY

55521

T0091.097776

60/029209

ADDITIONAL INVENTORS

Last Name	First Name	Middle Intitial	City and State
Abbott 200	Charles		Santa Monica, California
Boyd 300	Nik		Los Angeles, California
Cohen 400	Albert		Los Angeles, California
Cook 500	James		Manhattan Beach, California
Grandcolas 600	Michael		Santa Monica, California
Sikhun 700	Lan		Los Angeles, California
Lindsley 800	Bonnie Lindsley		Santa Clarita, California
Markarian 900	Grigor		Agoura, California
Moss 1000	Leslie		Los Angeles, California

55521

Limited Distribution

Requirements for NT Self-Service Delivery System Architecture

Version 1.0

Distribution of this document is limited to TTI employees.

The following are requirements for an NT Self-Service Delivery Architecture.

In defining the requirements, particular terminology is used. It will be helpful to understand this terminology before reading the requirements.

- Component - A component is any piece of a delivery system (hardware or software) that is potentially independently replaceable. Components have well defined interfaces. Software components are usually embodied as either executables (.EXEs) or dynamically loaded libraries (.DLLs).
- Application - A set of components that does a specific business function (e.g. Cash Withdrawal). An application is envisioned to typically consist of several components: one or more Dialog components which handle the user interface, one or more Business Rule components, and one or more Transaction components which handle the message interface with External Service Providers.

1. PLATFORM REQUIREMENTS

1.1 To evolve to a common application base for customer self service applications on all access devices.

1. CATs, CASSTs, PC CATs, and other kiosk terminals.
2. Home banking terminals through dial-up lines: customer PCs/MACs, PDAs, screen phones, Internet Browsers, Interactive TV.
3. Applications could run on staff terminals (but architecture may not cover everything needed for full staff applications; ex. entitlements, dual screens).
4. Applications could be moved to a new device types or environments with small incremental cost (e.g. IVR environment).

1.2 To support platform components from Citicorp selected standards:

1. Client: Application components run on iX86 platforms with Windows NT/95; User interface projectable to Macintosh and other Citicorp selected non-Windows platforms.
2. Server: Platforms running Windows NT

1.3 To support 3rd party devices and off-the-shelf software components.

1. Portability of applications and application services onto 3rd party platforms that comply with the architecture specified system platform standards.

2. Support of widely accepted industry standard hardware interfaces to enable integration of 3rd party peripheral components.

3. Compliance with industry standard software development techniques and interfaces to ensure integration of off- the-shelf 3rd party software components.

2. USER INTERFACE REQUIREMENTS

2.1 To support state of the art user interfaces customized for the environment.

1. An application can have different presentation styles on different device types (e.g. CAT, home PC, PDA, screen phone, etc.).

2. Support multiple languages.

3. Support style flexibility by region (color schemes, graphics, icons, font size, etc.) to satisfy local cultural differences.

4. Support variations for country/currency, customer type, and local business needs.

5. Allow easy integration of off-the-shelf multi-media elements into the user interface where supported by the display environment (e.g. sound, images, sophisticated graphics, video, animation).

6. Support multiple field input forms with "point-and-click" navigation between fields, and input data validation including basic field validations (re-usable) and cross field validations.

7. Support local dynamic changes at the client to form templates and data based on user selection or input.

8. There should be similar mechanisms for formatting printed output as for display output.

2.2 To support flexible navigation paradigms.

1. A user should be able to work at several application points simultaneously and switch between them. These could be in separate applications, or

multiple instances of the same application. One should be able to leave at any point in an application and start another application, while saving the original context and allowing a later switch back or return to it.

2. There should be a mechanism for exchanging user's data between applications, for example, so the user doesn't have to reenter data.

3. The architecture should support multiple types of user navigation modes to support multiple paths to an application, including for example: Linear (guide user through detailed Q&A steps as in current CAT), Non-linear broad branching (e.g. pull down menus), Preferred (e.g. user specified shortcuts), Query (e.g. search engine, natural language)

4. The architecture should support the customer being able to select his own top level navigation style (e.g. novice vs. expert modes).

5. The architecture should support both a step by step Question & Answer type dialog as well as a forms based input style.

6. The architecture should support the customer being able to define his own sets of transactions with predefined parameters that can be ' activated whenever he wants (e.g. normal "get cash", or "end of month" transactions).

3. APPLICATION ARCHITECTURE REQUIREMENTS

3.1 To ensure smooth migration from AGS to the new architecture.

Support gradual migration by means of the harmonious co-existence of applications built under the new architecture with existing AGS applications.

3.2 To support re-usable components.

Support convergence to a base set of re-usable components which can be easily and safely customized, enhanced, and re-assembled in different fashions by a local business.

3.3 Modular architecture that supports separation of top level navigation, user interface, business logic, and generic application services.

1. Should allow many-to-one interaction relationships between application components (e.g. several Dialog components for one Transaction component, or several Dialogs using the same Business Rule).

2. Should allow an application to be stand-alone or to be gotten to from another application (e.g. Transaction Journal could be stand-alone or gotten to from Account Summary).

3. Components should be able to operate independent of location, and be easily migrated between client systems and any intermediate servers in the path from the client to "front-end" servers.

4. Isolation of data-elements that need to be configurable in a local business (e.g. language specific phrases, product names, country data, currencies, regulatory and marketing messages, etc.).

3.4 To support flexible navigation paradigms.

1. Top level navigation separated from user interface and business logic. Should be easy to define by non-technical staff.

2. Top level navigation can be dynamically re-configured at runtime based on the existence or availability of devices, customer type and preferences, and applications available.

3. Support event driven application flows that give the user broad capability to jump around.

3.5 Support of "auto-configuring" applications that:

1. Auto-register and insert themselves into top level menus.

2. Adapt themselves to particular environments (e.g. devices present and their characteristics).

3.6 Provide a set of generic services for applications including:

1. Session management - startup, shutdown, shared data context across applications.

2. Top Level Navigation - selecting and switching between applications.

3. Device access/management - device interfaces (including 3rd party), standardized status & MIS reporting.

4. Data entry/display - selection elements (e.g. buttons, menus), forms handling, basic input data field validations (e.g. type, range, formatting checks), data display formatting.

5. Network services - communications transport, inter-application communication mechanisms; database access; mail access.

4. DEVELOPMENT AND TEST REQUIREMENTS

4.1 Application Development

1. Easy-to-use Prototyping / Development tools for top level Navigation Definition and User Interface design.

2. Application assembly using pre-fab components and templates instead of "from scratch" development.

3. User Interface design makes use of pre-defined Look & Feel standards.

4. Tools to support configuration management of software.

4.2 Testing

1. A stand-alone Debug/Test environment is provided for the developer's desktop with simulators for devices and external service providers.

2. Possible to test/certify components individually, or in combination where they interface to each other.

3. The ability to support automated un-manned testing of applications including regression, error, and performance testing.

4.3 Productivity

Tools support parallel development and testing.

5. OPERATIONAL REQUIREMENTS

5.1 Performance

Deliver adequate user response time over all communication links, including dial-up lines at 9600 baud and above.

5.2 Scalability

Systems can be scaled to support more users and access devices as needed, in a manner at least as cost effective as today.

5.3 Interfaces to current front-end service providers

Interfaces using their current and future GCB mandated standardized interfaces: TPS, SFS, RFS, GCIF.

5.4 Security

1. Provides integrity/privacy of customer data
2. Robust enough to protect transactions on GRN and Internet
3. Comply with CISO and Corporate Audit requirements
4. Authentication of staff (for admin & maintenance) across multiple systems
5. Mutual authentication of client/server
6. Audit trails

5.5 Configuration done by local business with simple user friendly tools.

1. Top level navigation
2. Applications run networked or stand-alone
3. Devices (e.g. types of bills in CDM)
4. Changes for product names, languages, currencies, messages (regulatory & marketing), product/account rules, etc.

5.6 System management

1. The ability to start/stop application sessions.
2. All applications instrumented for the following: status inquiry, performance monitoring
3. Instrumentation based on standard templates to ensure consistency across all systems.
4. Local/remote diagnostics.

5.7 Logging and Error Reporting

1. Common logging mechanism for runtime and applications.
2. Standardized logging formats across all systems.
3. Exception events reported to centralized trouble management system.
4. Sufficiently detailed exception reporting to identify problem.
5. Local and remote perusal of logs.

5.8 Software distribution/management

1. Distribution component based: initiated by central site for Citibank devices, initiated by server for customer devices with provision for customer to accept/decline.
2. Cutover/fallback of releases.
3. Versioning for software inventory.
4. Software validation (integrity checking).
5. Protection from running wrong type of software on a device.

5.9 CONSTRAINTS

1. The architecture applies to customer activated applications for Interactive Devices.
2. Applications get customer data through requests to external systems.
3. Fits within the GCTG architecture framework.

Limited Distribution

NT Self-Service Delivery System Architecture

Version 1.0

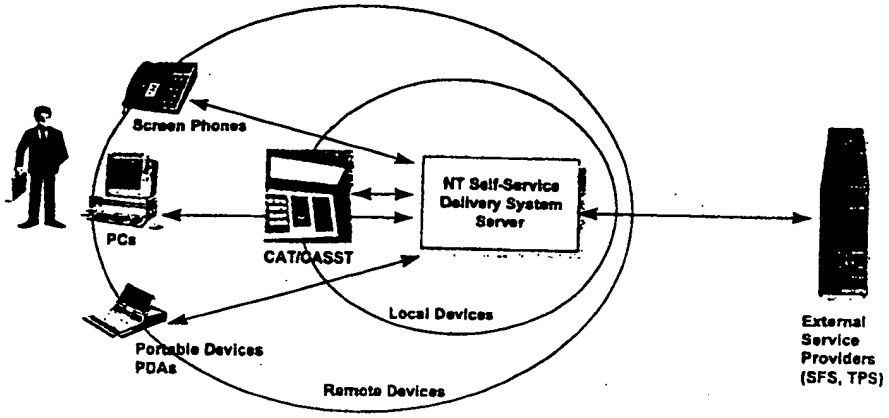
Distribution of this document is limited to TTI employees.

Table of Contents

1. Introduction

1.1 Focus, Scope and Context

Scope of NT Self-Service Delivery System Architecture



The primary **focus** of this document is the conceptual application architecture for the next generation of Citibanking applications. This conceptual architecture will focus on the high level components of the architecture and their interactions. Following the adoption of this conceptual architecture, the next level of design work will take place fleshing out the details of each high level component, their sub-components, interfaces and actual inter-component mechanisms that will be adopted.

The **scope** of delivery for these applications spans self-service terminals like CAT and CASST and remote customer devices like PCs, Screen Phones or PDAs. For customer devices these applications are to be delivered via both industry standard client software like Web browsers and custom Citibank client software.

Therefore the **context** for this architecture includes customer touch points (e.g. CAT Touchscreen or Web Browser), delivery networks (e.g. dial-in, internet, online services) and application servers (e.g. HSDS).

For this architecture we will leverage established and emerging industry standards and tools as well as domain specific knowledge gained over a twelve year period in delivery of Citibanking via CATs and HSDS.

1.2 Leveraging Industry Standards

Clearly the emergence of Internet Web browsers, servers, Java and the newly announced Microsoft Internet strategy, including changes to OLE in the ActiveX strategy are components and tools we can leverage in our application architecture.

In addition, Microsoft standard facilities for separation of components thru DLLs and component interaction via OLE will be widely used in this new architecture.

While the application functionality must be delivered to many types of devices, it is assumed that the operating system of the server, serving up these application services is Microsoft's Windows NT.

1.3 Leveraging Domain Specific Knowledge

The Home Services Delivery System's (HSDS) session and delivery model is more general than the CAT's, in that it supports local and remote delivery, multiple servers and capacity scaling. As such HSDS' session and delivery model contains a baseline model for the new delivery system architecture.

However, as will become apparent, many of the current HSDS services that are bundled together as a monolith, will be decomposed and exposed as self-contained objects or components. Other lessons learned in HSDS, especially with regards to Direct Access's separation of applications as separately callable functions, will become even more important in this architecture.

A detailed look at existing CAT applications instructs us in the need to put formal mechanisms around business rules, language handling and especially application isolation. The last point may be the most important of all. To the extent that this new architecture creates a framework where a new application can be safely and dynamically added to a suite of existing applications, without the need to regression test the entire suite, is the single most important thing we can do to reduce the time to market for new functionality.

We will also leverage the good work already accomplished in porting the CAT peripheral infrastructure to NT and exposing their interfaces as OLE components.

1.4 Objectives of the Application Architecture

- To evolve to a common application base for customer activated applications for all Touch Point devices on which they need to be provided, including:
  - ◊ Home banking devices (PCs, smart phones, Internet Browsers, PDAs)
  - ◊ CAT/CASST platforms
  - ◊ Branch and CSR staff platforms

- To support convergence to a base set of re-usable global application components that can be
  - ◊ re-assembled in different combinations and organizations to form application suites
  - ◊ customized for the environment in which they are used
  - ◊ complemented by components from a local business

- To provide state of the art user interfaces that support
  - ◊ integration of standard multi-media elements (pictures, video, audio)
  - ◊ customizations needed for specific devices, languages, countries, and other local business needs
  - ◊ multiple coexisting application navigation paradigms
  - ◊ the user working in multiple application components at one time

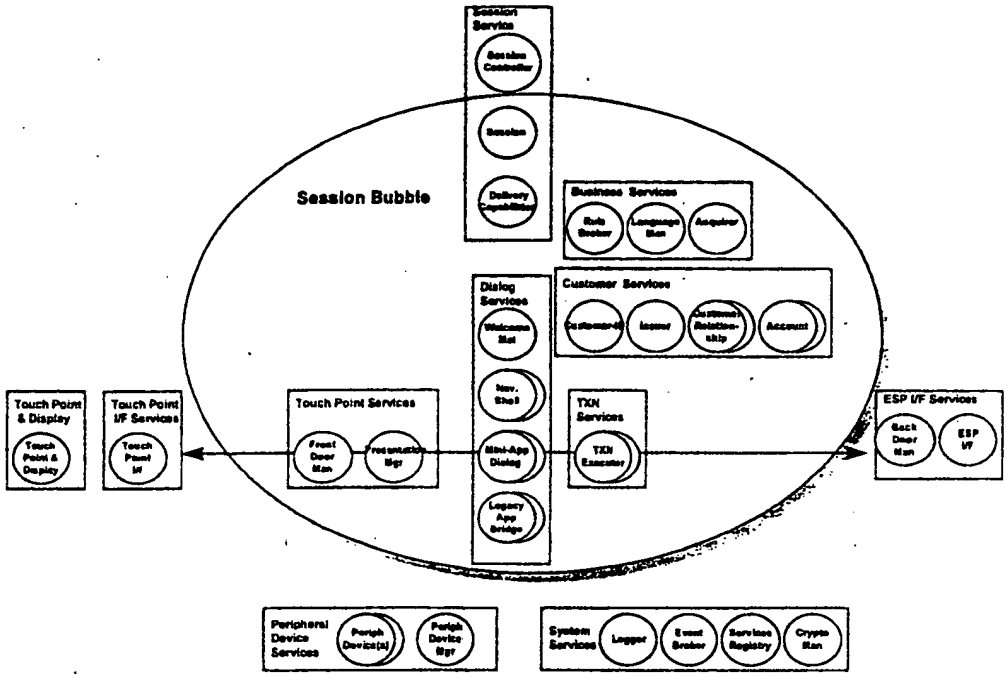
- To improve development and maintenance cycle time through
  - ◊ Application assembly using pre-fabricated components and templates instead of "from scratch" development.
  - ◊ Embracing widely accepted industry standards for component interfaces so that we can use off the shelf "plumbing" to connect components, and enable plugging in 3rd party components.
  - ◊ Support of "plug and play" application components that can automatically configure themselves for the environment, and automatically insert themselves into top level navigation menus.
  - ◊ High productivity Prototyping / Development tools for top level Navigation Definition and User Interface design, making use of pre-defined Look & Feel standards.
  - ◊ Separation of different parts of an application so that changes in one part don't affect the others:
    - * User interface, navigation, business logic, generic services.
    - * Language, country, currency, business specific messages.

- To provide a smooth gradual migration from legacy applications to the new architecture
  - ◊ By supporting the harmonious coexistence of software built under the new architecture with existing legacy AGS applications.

2. Conceptual Model Overview - A Set of Service Components

2.1 Overview

The conceptual model of the NT Self-Service Delivery System Architecture presents a set of service components.



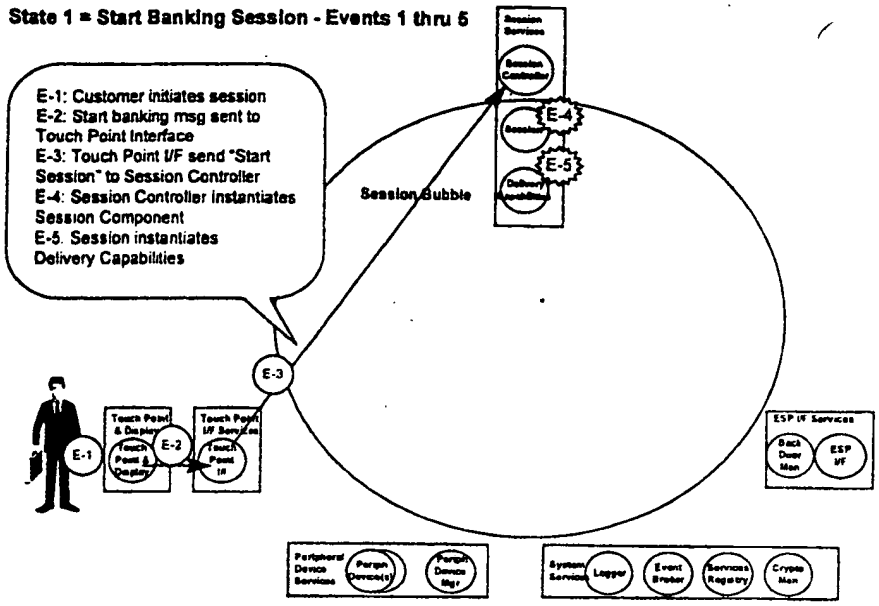
These service components fall into several broad categories:
Business Services: provides rule brokering, language and acquirer services.
Customer Services: provides customer id, relationship, account & issuer services.
Dialog Services: provides welcoming, navigation shell and app specific dialogs.
External Service Provider Interface Services: provides message sequencing and ESP I/F protocols.
Peripheral Services: provides peripheral device interface and management services.
Session Services: provides session startup and session and delivery vehicle context.
System Services: provides logging, event brokering, service registry and crypto services.
Touch Point & Display: provides the actual customer display and input facility.
Touch Point Interface Services: provides the interface to Touch Points (especially remote).
Touch Point Services: provides presentation mapping and front door security.
TXN Services: provides txn coordination and ESP message formatting.

Short hand description and examples of Service Components:

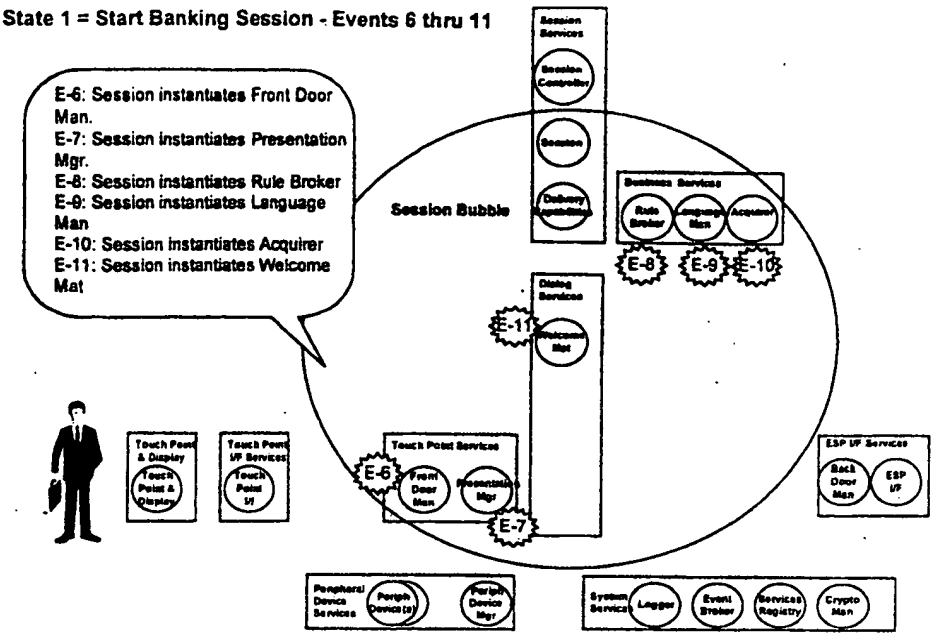
Account:- new component representing individual customer accounts (DLL or OLE)
Acquirer - new component representing general Acquirer business rules (non-Mini-App specific) (DLL or OLE)
Back Door Man - new component similar to some functions of HSDS' Common Integrator, responsible for message sequencing over all requests to a particular External Service Provider (DLL,OLE or EXE)
Crypto Man - new component responsible for key exchange, security services (DLL or OLE)
Customer-ID - new component that knows customer identification information (DLL or OLE)
Customer Relationship - new component that knows the customer's profile (DLL or OLE)
Delivery Capabilities - new component that knows the capabilities of the target delivery device (e.g. supports graphics, supports Pictographic fonts etc.) (DLL or OLE)
External Service Provider I/F - initially may reuse existing HSDS and CAT Gateway software
Event Broker - new component for event registering and dispatch (DLL or OLE).
Front Door Man - similar function that the Token Mapper in HSDS provides for guarding the front door of a session (DLL or OLE)
Issuer - new component representing general Issuer business rules (non-Mini-App specific) (DLL or OLE)
Language Man - new component, responsible for providing proper phrase resolution based on language and delivery vehicle (DLL or OLE)
Legacy App Bridge - component that wraps existing AGS runtime to allow AGS and New apps to coexist
Logger - wrapper component around NT logger
Mini-App Dialog - new decomposition of applications, into separate and independent mini-apps for particular functions (e.g. Pay a Bill) (DLL or OLE)
Navigation Shell - new component providing high level navigation for customer (DLL or OLE)
Peripheral Device Mgr. - component provided by new NT CAT infrastructure for device management.
Peripheral Device(s) - set of components provided by new NT CAT device interfaces (e.g. for depository, CDM) others provided by standard device drivers.
Presentation Mgr. - similar functions that the Token Mapper in HSDS provides for presentation (DLL or OLE).
Rule Broker - new component, central registry and router for business questions (DLL or OLE).
Session Controller - session startup and monitoring (EXE)
Session - new component providing (1) instantiation of initial session resources and (2) registry of session components brought up. (DLL or OLE)
Services Registry - wrapper component around NT Registry
Touch Point & Display - the actual customer interface (e.g. a Web Browser, Screen Phone, CAT TouchScreen & Display).
Touch Point Interface - interface software to customer touch point (e.g. Web Server, Prodigy Gateway etc.).
TXN executor - new decomposition of component responsible for one and only one transaction function to an External Service Provider(DLL or OLE).
Welcome Mat - new component does welcome dialog with customer, responsible for ID-ing the customer (DLL or OLE).

2.2 Example Session Walk-Thru

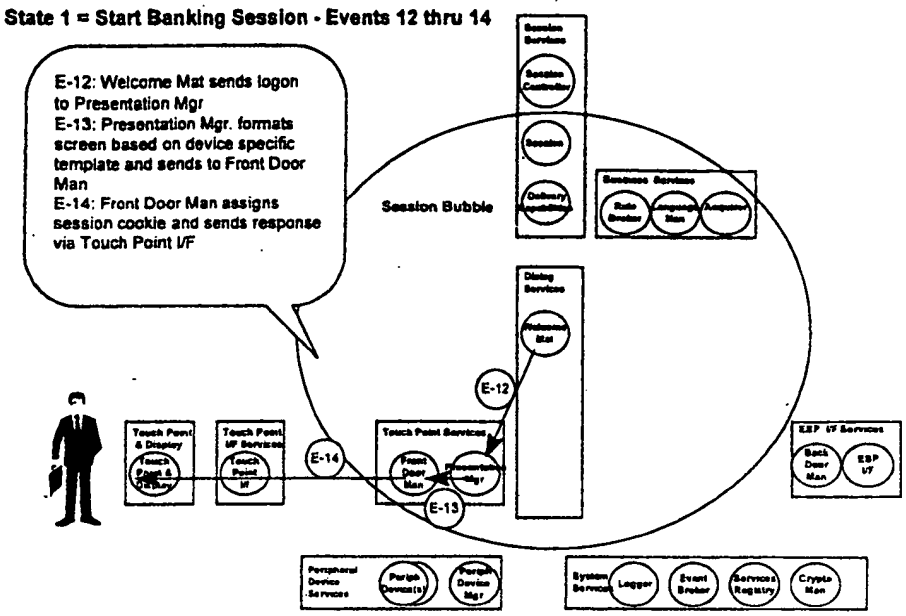
The simplest way to explain the conceptual architecture is to walk-thru an example session.



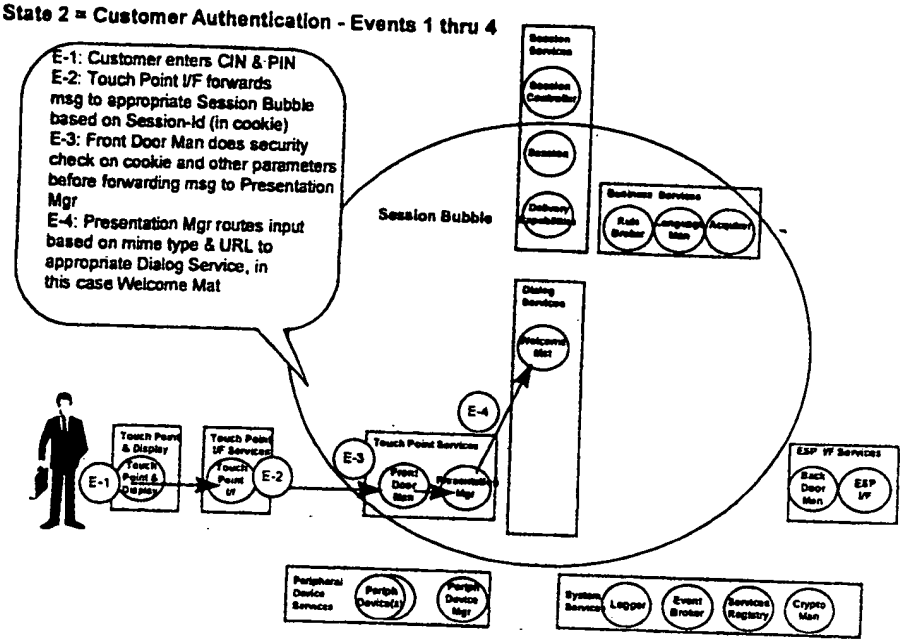
State 1 = Start Banking Session - Events 6 thru 11



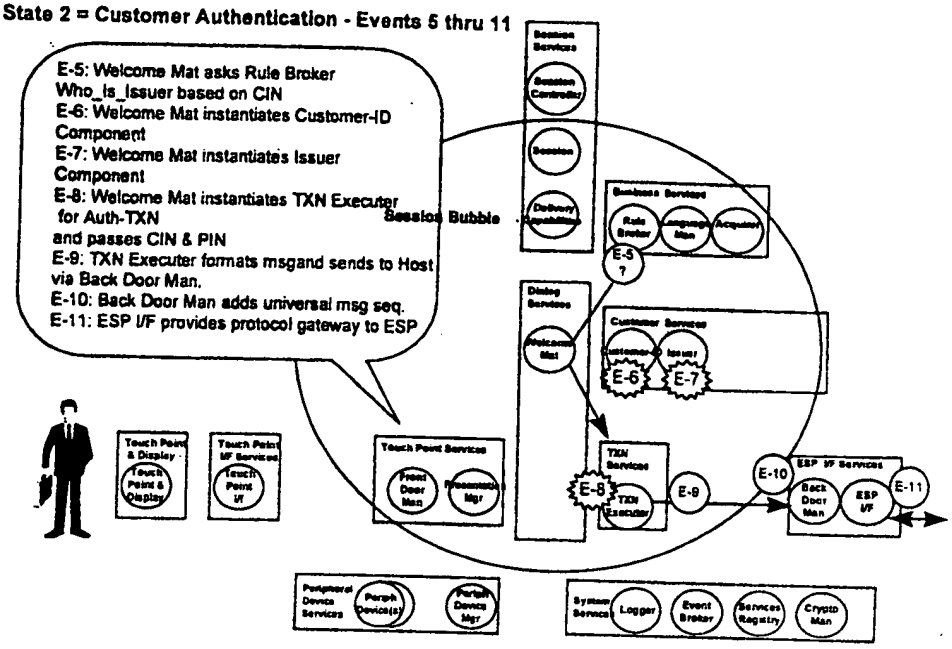
State 1 = Start Banking Session - Events 12 thru 14



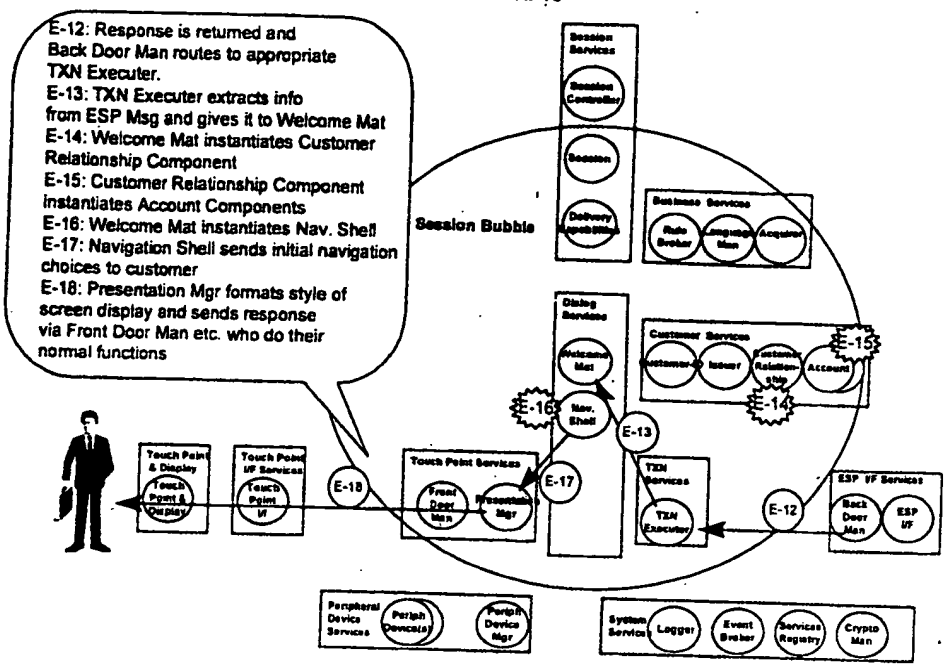
State 2 = Customer Authentication - Events 1 thru 4



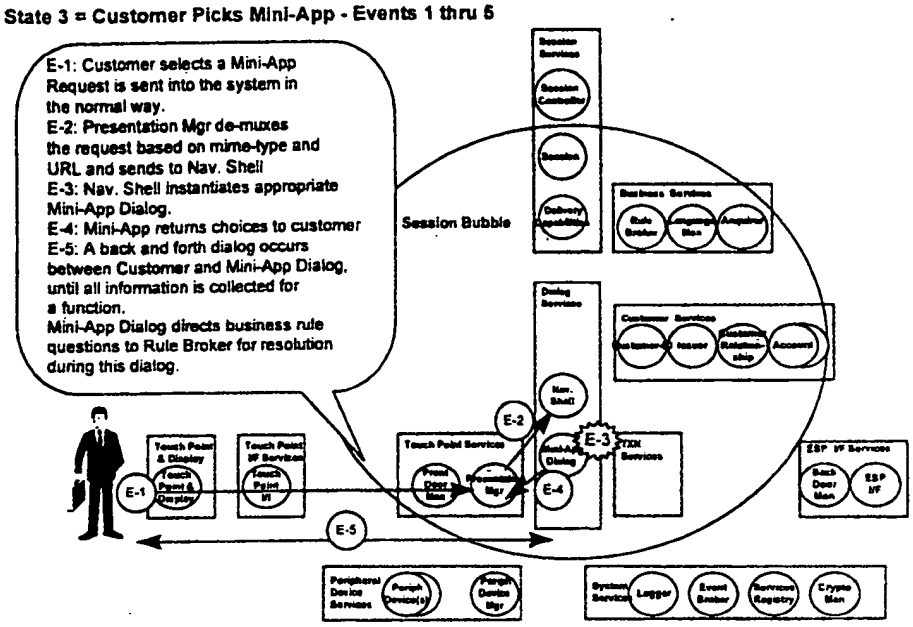
State 2 = Customer Authentication - Events 5 thru 11



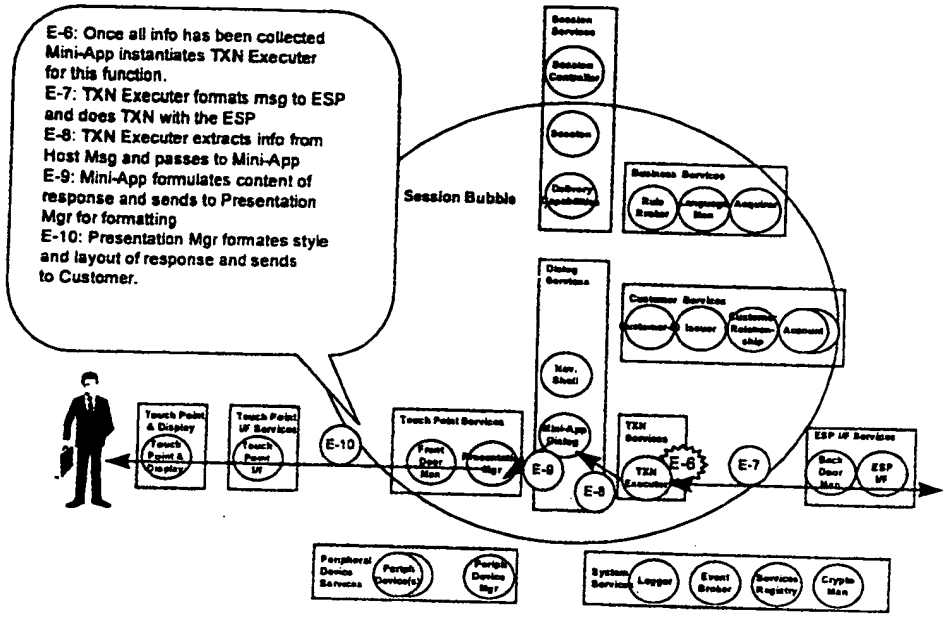
State 2 = Customer Authentication - Events 12 thru 18



State 3 = Customer Picks Mini-App - Events 1 thru 5



State 3 = Customer Picks Mini-App - Events 6 thru 10

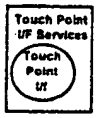


2.3 Mapping Service Components to Processes

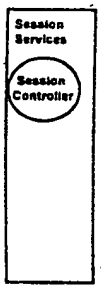
The following process models attempts to balance the tension between independence of components and efficiency at runtime. The detail design that will follow this architecture task, may refine the process mapping.

The following are the proposed process groupings:

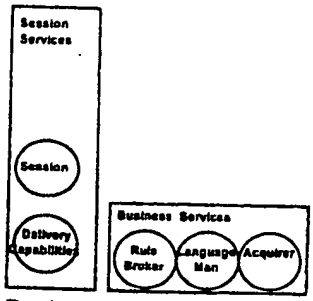
The Touch Point & Display is likely to be a separate, purchased component, especially for remote devices and even on the CAT. On Remote Devices examples include, Web Browsers, AOL Client Software, Screen Phone ROMs, PDA software etc. On the CAT the proposal is to use a customized Web Browser for this component.



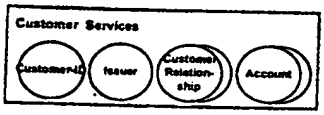
The Touch Point & Display Interface is likely to be a separate, purchased or operating system supplied component. Examples include Web Server software both for Remote Delivery and local delivery on the CAT.



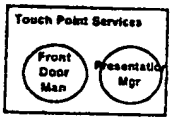
The Session Controller needs to be a self-contained process for the control and integrity of a specific NT Self-Service Delivery System Server.



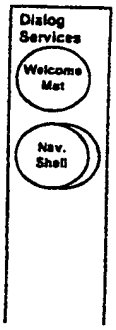
The Session Bubble specific Session Services and the Business Services functions seem to naturally group together. Both sets of services are needed for all session types and are more general in nature than other services within the Session Bubble.



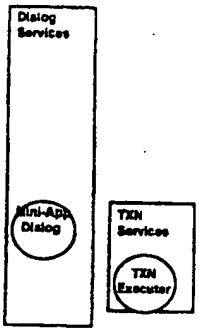
The Customer Services components of course group naturally and can easily coexist within a process.



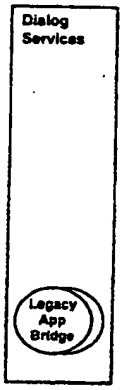
Experience with HSDS and Remote Projection have shown us that these two functions (currently embodied in the HSDS Token Mapper) group naturally as a independent process.



The Welcome Mat and Navigation Shell Components are likely to be delivery vehicle specific and therefore work naturally together and can be grouped in one process.



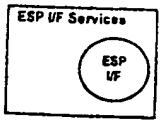
One of the key design goals of this architecture is to make every Mini-App independent of each other. The enforcement of making each Mini-App a separate process aids in this goal. The TXN executor is packaged with a Mini-App because it is likely each Mini-App will need a TXN executor. It is important to note however, that since TXN executor are separate components (implemented as DLLs or OLE) they can be repackaged with other Mini-App Dialog components.



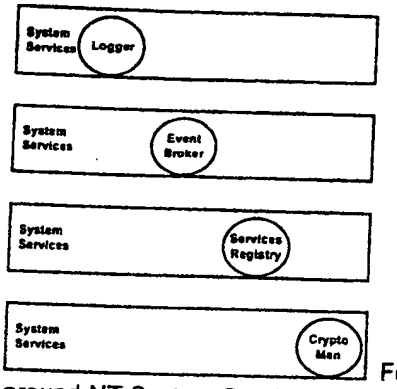
Legacy App Bridges, especially CAT AGS application sets are packaged as a process, so this grouping is a given.

The work already done in porting CAT peripheral support to NT is implemented as a process.

Since the Back Door Man must maintain External Service Provider message sequencing independent of the physical connection to an ESP it is separated out as a single component. Since it provides services to all sessions, not just one it needs to be a non session specific resource also.



Initially the External Service Provider Interface Components will be provided by existing Gateway code in HSDS and the CAT. This makes it necessary to separate them out as individual processes.



For the most part these components are wrappers around NT System Services components, packaged in separate processes.

2.4 Mini-App Packaging Model

A fundamental principle of this architecture is the independence of one Mini-App from another. The goal is to produce a safe environment for the dynamic insertion and registration of new Mini-Apps with their Navigation Shells. The goal is to do this in a way that the introduction of a new Mini-App entails a complete testing cycle only on the introduced application not a regression test of the entire system. To that end the packaging of Mini-Apps as separate entities is important.

The elements of a Mini-App Package are the following:

1. The Executable (.EXE) for the Mini-App Component, including the TXN Executor component as a DLL or OLE object.
2. A RULE file including all new Rule Entries to be registered with the Rule Broker.
3. Where appropriate, a Rules Engine file per rule, for any rules that can be interpreted by the general purpose Rules Engine.
4. Where appropriate, a Rule Database file per rule, that supplies any needed data to support Mini-App specific rule authorities.
5. A Language file including all Mini-App specific language phrases needed.
6. Where appropriate, a Template file containing all Mini-App specific templates.

2.5 Rendering Model

Basic Model for Rendering is indirect

To allow for both local delivery to a CAT and remote delivery to PCs, NetCAT, etc. the basic rendering model is indirect. This means that none of the Dialog Services components draw directly to the screen. Instead they produce a stream of data called the *App Stream* that will ultimately be rendered by Touch Point & Display devices or software.

The App Stream is HTML

The App Stream is an HTML encoded stream of named objects (or tokens) within named templates (or forms). The Dialog Services components (i.e. Welcome Mat, Navigation Shell and Mini-App Dialogs) are responsible on output for setting the properties of these named objects within named templates.

Dialog Services Components responsible for Content

While a Dialog Services component can set any property of a named object, the intent of the architecture is to encourage separation of content from style so that a specific Mini-App can be leveraged and delivered across many delivery vehicles.

In general, the Mini-App component will operate by setting the values of named properties of named objects in named templates. For example, TemplateX.ObjectY.PropertyZ = Value.

Presentation Manager Component responsible for Style

The Presentation Manager, using delivery vehicle specific named templates is responsible for style and mapping to the encoding language of the target device. The Presentation Manager takes the App Stream received from Mini-Apps and based on delivery vehicle specific templates merges the data based on mapping rules and produces the final token stream that is sent to the Touch Point & Display.

Delivery Vehicle Templates define Style

There is a one to one mapping between Canonical templates that Mini-Apps reference and Delivery Vehicle Specific Templates that the Presentation Manager uses. Delivery Vehicle Specific Templates include specific information on the layout, colors and mapping of individual objects. There are a set of emerging standards from Microsoft and WC3 on advanced style sheets, including style sheets that allow precise X,Y

positioning of objects that will be examined for use in the design phase of this architecture as the templating mechanism.

Separation of Content from Style

Separation of Content from Style provides many benefits. First it allows Citibank to define and change the style and layout of a presentation independent of the code in the Mini-Apps. This is essentially the page maker approach used in publishing where a newspaper or magazine has a certain style defined by its layout templates. Second it allows a single Mini-App to deliver its functions to more than one target delivery vehicle thru the abstraction of individual objects or tokens.

Abstraction of Objects or Tokens

This architecture encourages the use of abstract objects in the App Stream. For example the use of an abstract object like "choice" instead of a specific object like "button" allows a choice to manifest itself in many ways on the target delivery vehicle. A choice could manifest itself in one case as a CAT button, in another as a Windows style button, or a HTML anchor or an item in a scrolling list.

Use of ActiveX Controls

The new architecture will also support the inclusion of new ActiveX Visual controls within Delivery Vehicle Specific templates. However effort will also be expended in trying to map these controls to alternative objects for presentation on delivery vehicles that don't support ActiveX controls.

Grouping of Objects

This architecture also encourages the grouping of logically related named objects into named groups. This grouping facilities allows the masking out of a group, so that it wouldn't be delivered to certain delivery vehicles based on the capabilities of that device or screen real estate etc.

Presentation Model - Templates, Frame Sets, Frames

Delivery Vehicle Specific Templates define layout and style both for Frame Sets and within a Frame. A Frame is a well known concept within Web Browsers. A Frame is a rectangular portion of screen real estate (bordered or borderless). A Frame Set defines the layout of Frames within an overall screen window. The Frame Set defines the width and height of each frame and an initial link to the HTML page (or program) that will provide the content for that Frame.

The Presentation Manager manages the overall display. Based on Templates it

assigns a Frame or Frames to a Navigation Shell Component. In turn based on Templates the Navigation Shell assigns a Frame to a Mini-App Dialog. Within a Frame the layout of that Frame is controlled by a Delivery Vehicle Specific Template. Assigning frames that bound the display space of specific Mini-App Dialogs will maintain an independence between one Mini-App and another and also allow different Navigation Shells to be installed independently of the Mini-Apps.

Internally the Presentation Manager will model the display space as a set of frames. Based on the Delivery Vehicle Specific templates for non-frames devices the Presentation Manager will merge information from many frames (and or exclude frames) into a single frame for delivery to the device.

Accommodating Legacy Apps within a Frame

The HouseCAT project has already accomplished embedding a NAPLPs CAT stream within an HTML page. We will use either that approach or the HTML tokenization of CAT AGS legacy applications to place the legacy CAT application display within an overall display space that includes a Navigation Shell as well as new Mini-Apps.

Links and Input Data

The Canonical Templates that Mini-Apps use are bounded by a Frame. As was stated earlier, the Mini-App is responsible for setting the properties of the named objects within its canonical templates. One of the properties the Mini-App is responsible for setting for "choice objects" is a link. A link is a standard Universal Resource Locator (URL). The URL specifies the target object (in this case the Mini-App) and a list of parameters that will be returned if this choice is selected by the customer. The activation of links by the customer is one of the main ways of making choices and navigating thru a Mini-App Dialog.

Besides links, input data entered by the customer in input fields, select lists, check boxes, radio buttons etc. will be returned to the Mini-App in the App Stream in the standard HTML encoding style of Name-Value pairs.

Basic App Stream Interface is Easy to Produce.

Any programming language that can produce a text stream can produce a App Stream. Coupled with the requirement that Mini-Apps need to communicate via DLL or OLE, this implies that any language capable of those three things can be used to write a Mini-App.

Advanced Object References in the App Stream

The App Stream is in fact a multi-channeled stream capable of supporting the Basic Text Based App Stream as well as other mime types. Investigation of emerging tools

from Microsoft may also provide us with the ability to pass references to named templates (forms) as objects from a Mini-App to the Presentation Manager. This capability will be investigated during the design phase.

Delivery Vehicle Specific Dialogs

While the architecture encourages leveraging one Mini-App over a large range of delivery vehicles, it does not preclude writing Mini-Apps targeted towards a specific delivery vehicle or class of delivery vehicles. The mechanism of passing the App Stream between Mini-Apps and the Presentation Manager remains the same. The Mini-App is still responsible for content and the Presentation Manager for style and layout. However in this case, the range of visual object types or capabilities may only be available on a specific delivery vehicle and doesn't lend itself to abstraction. For example the inclusion of client-side scripting may only be available on certain devices or class of devices and could not be easily abstracted.

Support for Multi-Media

HTML has well-known means for embedding and referencing a wide range of media types, e.g., graphics, sounds, and movies. The architecture will use standard HTML encoding techniques to incorporate this ever-expanding set of media types into the system for use by the Touch Point Services and Dialog Services components.

Caching of Output

To support various error conditions and easy switching and restarting of Mini-Apps, the Presentation Manager component will cache the last page output for each frame that it manages.

2.6 *NetCAT in the NT Self-Service Delivery System Architecture*

NetCAT provides the ability to present a traveling customer their "home screens". This is accomplished without the need to load all regions CAT software on all CATs around the world. The basic notion is to have at least one NetCAT Server for every region. On this NetCAT Server a region's CAT software will run and it will be capable of being "remotely projected" thru any acquiring CAT around the world, thus providing almost all of a customer's "home screens" around the world. Differences will show up on the initial Welcome Screen, until the customer's issuer is identified, and during certain transactions (notably cash withdrawal) where foreign exchange rates will have to be displayed and regulatory requirements of the acquiring country will have to be honored.

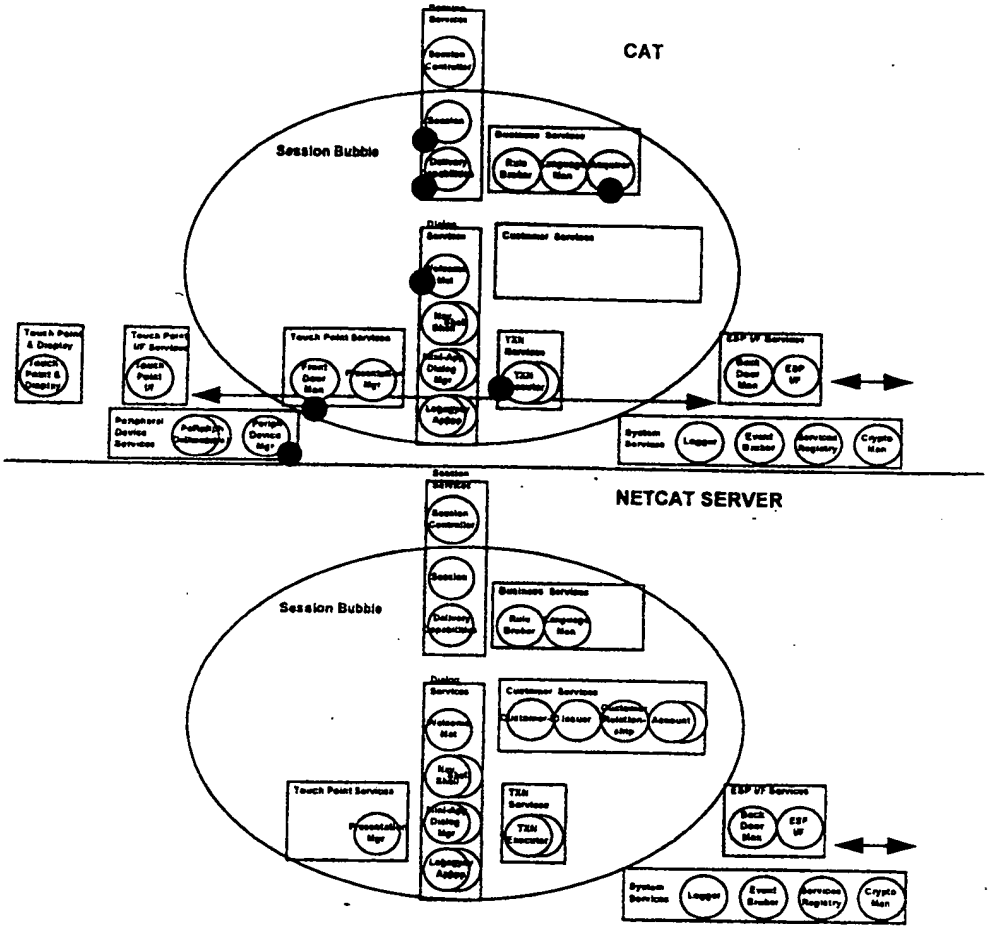
Starting a NetCAT Session

The traveling customer dips card at a "foreign" CAT. A session bubble starts up normally at the CAT. When the Welcome Mat determines this customer is off-region, it makes a connection to the appropriate regional NetCAT Server. Welcome Mat on the CAT, communicates with the Session Controller on the NetCAT Server to start up a Session. The Welcome Mat on the NetCAT Server, given card parameters passed to it on startup, instantiates the Customer-ID and Issuer components on the NetCAT Server. After NetCAT server authenticates the customer, with its own External Service Provider, it starts up a Navigation Shell component on the NetCAT Server.

The CAT exposes/copies certain of its components to the NetCAT Server for use. Specifically the CAT exposes the following components marked with a black dot on the diagram:

- Session
- Acquirer
- Delivery Capabilities
- Front Door Man
- Peripheral Device Manager
- TXN Executor (specifically the Cash Withdrawal one)

The NetCAT server uses these components for business rule inquiries, for delivery to CAT screen, for operation of the CAT peripherals and to inquiry about the capabilities of the hosting CAT (e.g. fonts supported, pictographic printing etc.).



3. Major Service Components Described - Roles & Responsibilities

3.1 Session Services

At the heart of the system is a set of Session Services. The components within the Session Services category are:

Session Controller
Session
Delivery Capabilities

3.1.1 Session Controller

The Session Controller is responsible for managing all the sessions in the server. When a new customer contacts the server, the Session Controller starts a session by instantiating a Session Bubble for the session. The Session Bubble bounds a secure set of resources allocated to one and only one customer session. The Session Controller is aware of the type of customer Touch Point a start session request came from and the broad product type of service requested, so that the appropriate type of Session Bubble can be instantiated.

Responsibilities:

- Creates a session when a customer contacts the server by instantiating a new instance of the Session object.
- Maintains a registry of all active sessions with handles to the Session objects.
- Terminates a Session when a customer abnormally breaks the connection.

Instances: One per server.
Uses/Calls: Session
Used by: Touch Point I/F (when a new customer contacts the server).

3.1.2 Session

The Session component manages the resources associated with a session. It has two important functions (1) to bring up some initial session resources and (2) to be the registry for session components that are brought up. The Session component also knows certain session context information as well as all assigned session resources and services.

Responsibilities:

- Instantiates the following resources when a session is created, and deletes them when the session is terminated.
  - ◊ Front Door Man
  - ◊ Delivery Capabilities
  - ◊ Presentation Manager
  - ◊ Acquirer
  - ◊ Rule Broker
  - ◊ Language Man
- Starts Welcome Mat to authenticate customer.
- Tracks session states/events (e.g. Navigation Shell launched, Mini-App Dialog launched)
- Recovers resources when a session abnormally terminates.
- Logs significant session events (e.g. start/end of session, session errors).

Knows: Session context
- Session initiation information (Session ID, start of session time)
- Handles for linking to all other session resources
- Session languages (Presentation, Printer)
- Navigation Shells & Mini-App Dialogs active
- Application related device information (e.g. Printed record queue status)
- End of session reason (for end session MIS logging)

Instances: One per session.
Uses/Calls: Welcome Mat
Logger
Event Broker
Used by: Session Controller

3.1.3 Delivery Capabilities

This is a component that holds data and answers questions about the delivery capabilities of the Touch Point for a particular session. This information was communicated either explicitly or implicitly in the startup message from the device that caused the initiation of a session. The Delivery Capabilities component is available for interrogation from other components in the system.

Responsibilities:
- Answer questions about the delivery capabilities of a Touch Point (e.g. for a Web Browser Touch Point - HTML level, ftp, picture formats, applet types, script types, international fonts, etc.)
- Instantiated by Session, with initial capabilities based on access mode (e.g. Internet, dial-in, CAT).

Instances: One per session.

Uses/Calls: None
Used by: Presentation Manager
Dialog Services components (Welcome Mat, Navigation Shell, Mini-App Dialog, Legacy App Bridge).

3.2 Dialog Services

The components within the Dialog Services category are responsible for the semantic content and interaction with the customer and for initiating transactions on the customer's behalf. The components within the Dialog Services category are:

Welcome Mat
Navigation Shell(s)
Mini-App Dialog(s)
Legacy App Bridge(s)

3.2.1 Welcome Mat

The Welcome Mat is responsible for outputting the initial welcome page to the customer, and collecting customer identity and preference information. After determining the Issuer of the Customer ID, and possibly authenticating the customer, the Welcome Mat instantiates several Customer Services objects to hold information about the customer, and then starts a Navigation Shell which carries out the next level of dialog with the customer.

Responsibilities:

- Presents a welcome page (based on Delivery Capabilities).
- Collects customer preferences:
  - ◊ Language choice
  - ◊ Other preferences (e.g. navigation style)
- Collects Customer-ID information (e.g. CIN/PIN, Public Key Certificate) in a manner consistent with the customer Touch Point and mode of access (e.g. dial-in, Internet).
- Handles retries if errors occur on customer identity input (e.g. re-read card).
- Asks the Rule Broker for an Issuer Rule Authority based on the card/CIN prefix number.
- Calls a TXN Executor to authenticate the customer and get his relationships (customer profile). This typically involves interactions with the issuer external service provider, but in the future it may be done locally from information in a SmartCard.
- Instantiates the following Customer Services components:
  - ◊ Customer-ID
  - ◊ Customer Relationship and Accounts

  - ◊ Issuer (different issuers based on rules about Customer-ID)
- Starts a Navigation Shell (based on Delivery Capabilities, Acquirer Rules, and customer preferences)

Instances: One per session.
Uses/Calls: Presentation Manager (to output pages)
Delivery Capabilities
Customer ID
Customer Relationship and Accounts
Acquirer, Issuer
Rule Broker
Language Man (to construct phrases)
TXN Executor (authenticate customer and get relationships)
Navigation Shell
Used by: Session

Notes:
A Welcome Mat may use separate Mini-App Dialog sub-components to do some parts of the dialog that may be common to several business functions or which may vary depending on the Touch Point peripherals (e.g. card re-read, PIN entry).

If the Rule Broker does not have a registered Issuer for the card/CIN prefix number, a Customer-ID is instantiated and marked invalid, further authentication of the customer is skipped, and a Navigation Shell for invalid customers is started. Invalid customers may still be allowed to use certain information only Mini-App Dialogs.

The Welcome Mat may do four things for customer authentication based on Acquirer rules and the type of customer ID (Public Key Certificate, ATM card, credit card, on-us, off-us):
- Immediate local authentication (using Public Key Certificates).
- Immediate authentication with the Issuer, waiting for response.
- Background authentication with the Issuer, while going on to the Navigation Shell.
- Defer authentication to the first transaction.

In the case of deferred authentication, it may be necessary to instantiate a default Customer Relationship and a default set of account product types (checking, savings, credit, etc.)

3.2.2 Navigation Shell

The Navigation Shell is responsible for making the customer aware of the range of Mini-Apps that are available, and providing Top Level navigation across them.

The Navigation Shell assigns a frame space for a Mini-App to run in. To support complex grouping of functions or a variety of navigation styles there can be Shells within Shells.

There may be several Navigation Shells selectable by the customer to support different navigation styles, including for example:

- Linear (guide customer through detailed Question & Answer steps)
- Nonlinear broad branching (e.g. pull down menus)
- Preferred (e.g. customer specified shortcuts)
- Query (e.g. search engine, natural language)

Responsibilities:

- Makes the customer aware of the range of Mini-Apps available (based on the customer's relationship, and Issuer/Acquirer rules, and the set of dynamically registered Mini-Apps). The way Mini-Apps are organized and identified is Navigation Shell dependent (may be names, icons, etc.)
- Instantiates Mini-App Dialogs as requested by customer.
- Supports switching between concurrently active Mini-App Dialogs.

Instances: One Main Navigation Shell per session, possible layered Sub-Shells.

Uses/Calls: Language Man (to construct phrases)
Peripheral Device Manager (possibly to find devices up/down)
Presentation Manager (to output pages)
Services Registry (to find Mini-App Dialogs)
Mini-App Dialog(s)
Legacy App Bridge(s)

Used by: Welcome Mat

Notes:
In an environment where the screen has embedded frames, a Main Navigation Shell may invoke one or more Sub-Shells to control the individual frames.

The architecture supports the customer leaving a Mini-App to go to the Navigation Shell and start another Mini-App, while leaving the former Mini-App suspended in its current context state. The customer can later exit from the new Mini-App and go back to the former Mini-App, or he can switch between multiple concurrently active Mini-Apps.

3.2.3 Mini-App Dialog

The Mini-App Dialog manages the dialog with the customer for a specific business function (e.g. Transfer, Bill Payment) in a specific dialog style (e.g.

Q&A, Form, etc.). It presents information and choices to the customer, and collects and validates customer inputs. It is responsible for the content of information on pages and the flow of the customer interaction, but not the style and layout of the presentation.

There may be several different Mini-App Dialogs with different dialog styles for the same business function. These could support different modes of the customer entering information, such as:

- Guide the customer through detailed Question & Answer steps.
- Forms with multiple input fields.

After collecting the necessary customer inputs for a particular business function, the Mini-App Dialog uses a TXN Executor to carry out the function by doing transactions with external service providers and operating Peripheral Devices (e.g. cash dispenser, depositor).

Responsibilities:

- Implements the customer-visible control flow for a particular function in a specific dialog style. The flow may be tailored based on the customer relationship and various country/business rules.
- Uses the Language Man to do translation of phrases into target languages (display or print).
- Assembles phrases and formatted data into pages (display or print). A page is constructed in a canonical format by setting properties of named objects within named templates.
- Sends pages to the Presentation Manager which handles the final style and layout for the specific Touch Point device.
- Collects customer inputs.
- Validates customer inputs using business rules. Validation includes basic field validations and cross field validations.
- Calls TXN Executors to interact with external service providers and operate Touch Point devices (e.g. cash dispenser, depositor).

Instances:	Many per session (serially and concurrently)
Uses/Calls:	Presentation Manager (to do device specific formatting and page layout for both display and printed pages)
	Peripheral Device Manager (to get device statuses)
	TXN Executors (to do transactions)
	Rule Broker (to evaluate business rules)
	Language Man (to construct phrases)
	Customer Relationship & Accounts (to get account information)
	Issuer (to find services allowed, and evaluate Issuer Rules)
	Acquirer (to find services allowed, and evaluate Acquirer Rules)
Used by:	Navigation Shell

Notes:
A Mini-App Dialog may use separate Mini-App Dialog sub-components to do some parts of the dialog that may be common to several business functions (e.g. PIN entry, account resolution, enter currency amount).

3.2.4 Legacy App Bridge

This component is a bridge that enables a legacy application set to operate in the new architecture framework. The main purpose of the bridge is to do translation of data between Customer and Business Services objects in the new architecture and the form that this data is stored in the legacy applications. There will likely be a different bridge for each type of legacy application set (e.g. US CAT, Asia CAT, Latin CAT, Euro CAT).

Responsibilities:

- On entrance to the legacy application, it obtains data from the Session and Customer Services objects and translates it into the global data structures needed by the legacy application.
- On exit from a legacy application, it takes modified data from the legacy structures and puts it back to the Customer Services objects in the new architecture.
- Translates legacy pages into the canonical page structures needed by the Presentation Manager.
- Interfaces with the Back Door Man to send messages to external service providers.
- Interfaces with the Logger for logging errors and transactions.
- During initialization of the bridge component, it may need to interrogate the Rule Broker and various Rule Authorities (primarily Acquirer and Issuer) to obtain data needed to populate static tables used by the legacy applications for processing rules.

Instances:	One per type of legacy application per session.
Uses/Calls:	Presentation Manager
	Back Door Man
	Customer Services objects
	Logger
Used by:	Welcome Mat
	Navigation Shell

Notes:
Depending on how far the migration has gone, there may be several possibilities for the relationship between Legacy App Bridges and the Navigation Shell.

- The Navigation Shell provides the Top Level navigation across the new Mini-App Dialogs as well as the individual Legacy App Bridges.
- For some card types and Issuers, the Navigation Shell may be faceless and all business functionality is provided by the Legacy Apps. In that case Top Level navigation is provided within the Legacy Apps.

For CAT applications one of a pool of CAT TAFE runtimes will be assigned to a session at startup. The legacy applications will be assigned a frame space by the Navigation Shell to "play" their applications inside of. Individual CAT level 3 functions will be individually registered and exposed. The Navigation Shell supports exposing individual CAT level 3 functions without the need to traverse the existing Level 2 Menu structure.

Refer to the Legacy Migration section for more details.

3.3 Transaction Services

The components within the Transaction Services category are responsible for handling external service provider transactions and device operations needed to accomplish particular business functions. These components provide transaction coordination and external service provider message formatting.

3.3.1 TXN Executor

Each TXN Executor performs a particular business function (e.g. cash withdrawal) by doing transactions with external service providers and operating Peripheral Devices.

Responsibilities:

- Validates properties (data) obtained from Mini-App Dialogs to see that it has the required information needed to do the transaction. If not, it immediately returns an error.
- Collects additional information needed to do the transaction from other objects (e.g. Customer ID, Acquirer, Issuer, Account).
- Formats messages to be sent to external service providers.
- Orchestrates complex transactions by sending messages to multiple service providers (serially or concurrently) as needed.
- Parses response messages and returns information as properties of a TXN object.
- Operates Peripheral Devices needed to carry out the function (e.g. dispense cash).
- Recovers from transaction or device failures. May reverse transactions during recovery.

Instances: Many per session (serially and concurrently)

Uses/Calls: Rule Broker (to validate input data)
Back Door Man (to interact with external service providers)
Peripheral Device Managers (to operate devices)
Logger (to record transactions)
Event Broker (to notify of transactions)
Used by: Welcome Mat
Mini-App Dialogs
Other TXN Executors
Acquirer (to download Acquirer data)
Issuer (to download Issuer data)
Accounts (to get account information)

3.4 Touch Point Services

The components within the Touch Point Services category are responsible for final device specific presentation layout and front door security. The components in this category are:

Presentation Manager
Front Door Man

3.4.1 Presentation Manager

The Presentation Manager is responsible for mapping a canonical representation of information on pages into a specific style layout in a device specific presentation format. Thus the same application can have different presentation styles on different device types (e.g. PC, PDA, screen phone, CAT, 3rd party kiosk terminal, etc.). The style templates can also be customized by region to support local cultural differences in areas such as color schemes, graphics, icons, font sizes, etc.

Responsibilities:

- Maps tagged phrases and data from the application into specific fields of a particular page template referenced by the application. A template controls the layout and representation of multimedia elements, choice and data fields, and input forms on the page for a specific style and device type.
- Encodes the resulting page in the Touch Point device specific format, and sends it to the Front Door Man.
- On incoming messages from the Touch Point it converts choice information and form fields from the device specific format to a tagged canonical representation, and routes it to the appropriate Dialog Services component.

Instances: One per session.
Uses/Calls: Front Door Man (to interface with the Touch Point)

Used by: Dialog Services components (Welcome Mat, Navigation Shell, Mini-App
Dialog, Legacy App Bridge).

Notes:
The Presentation Manager uses delivery system specific templates to enforce consistent layout style across pages having similar choices, data fields, and forms. A template can be the superset of all possible objects on a page since the Presentation Manager can "drop out" fields and choices for which there is no data.

3.4.2 Front Door Man

Responsible for guarding the access of the Touch Point into a session.

Responsibilities:
For remote sessions:
- Adds a session security token to outgoing messages.
- Verifies the session security token for incoming messages.

For CAT/CASST:
- It may be non-existent or a pass through.

Instances: One per session.
Uses/Calls: Device Interface Manager
Used by: Presentation Manager
TXN Executors associated with Mini-App Dialogs

Notes:
There is a different type of Front Door Man for each Touch Point device type.

3.5 Touch Point Interface Services

This component provides the interface to the Touch Point (especially remote ones).

3.5.1 Touch Point Interface

Responsible on the server side for managing the link/session level protocols with a Touch Point.

Responsibilities:
- Notifies the Session Controller to start a new session, on initial contact from a remote Touch Point it
- Encodes messages in the Interface protocol.
- Sends messages to the Touch Point.

- Decodes messages received from the Touch Point.
- Routes received messages to the appropriate session Front Door Man.

Instances: One per touch point interface type.
Uses/Calls: Touch Point & Display
Used by: Front Door Man

Notes:
For Internet sessions this is a Web Server, which handles the protocols TCP/IP, HTTPS, ftp, etc.

3.6 Touch Point & Display

This component provides the actual customer display and input facility on the Touch Point.

3.6.1 Touch Point & Display

Displays pages on the Touch Point screen, and sends customer input to the application server.

Responsibilities:
- Manages the link/session level protocols with an application server, on the Touch Point side.
- Decodes the server interface protocol.
- Outputs a page to the local screen.
- Acquires customer input (choice selections and forms input).
- Encodes customer input in the server interface protocol.
- Sends customer input to the server (where it is received by the Touch Point Interface).

Instances: One per Touch Point.
Uses/Used by: Touch Point Interface

Notes:
For Internet sessions this is a Web Browser, which handles the protocols TCP/IP, HTTPS, ftp, etc.

3.7 External Service Provider Interface Services

The components within this category provide protocol support for interfacing with External Service Providers. The components within External Service Provider Interface Services category are:

Back Door Man

External Service Provider Interface

3.7.1 Back Door Man

This component is responsible for multiplexing messages from multiple TXN Executors in several sessions to a single external service provider. It provides message sequencing over all messages sent to a particular external service provider. It also provides response routing back to the requesting TXN Executor.

Responsibilities:

- Secures messages exchanged with an external service provider (MAC, encryption).
- Generates sequence numbers and adds external service provider envelope to outgoing messages.
- Sends outgoing messages to the External Service Provider Interface Manager.
- Responsible for retry of messages.
- Checks sequencing of incoming messages.
- Routes response messages to the proper TXN Executor.
- Routes incoming unsolicited messages to a registered or well known system component.
- Responsible for switching between alternate (or backup) external service providers to provide error recovery, load sharing, or alternate routing.
- Can support multiple outstanding requests simultaneously.

Knows:

- Which of the alternate (or backup) external service providers is active.
- Names/Addresses of external service providers.
- Server ID information (or knows how to get it).
- Message sequence numbers (in, out).
- Message security context.

Instances: One per external service provider
Uses/Calls: External Service Provider Interface Manager (send/receive messages)
Crypto Man (for message security)
Session (incoming unsolicited messages)
Used by: TXN Executors

Notes:
This component is logically similar to the Common Integrator in the CAT.

3.7.2 External Service Provider Interface Manager

This component provides protocol support for connecting to an External Service Provider. For example this component might provide X.25, 3270, or SNA protocol support. Initially this service will be provided by leveraging existing gateway components in HSDS and the CAT.

Responsibilities:
- Provides protocol support for a specific type external service provider interface if needed.

Instances: One per external service provider interface type
Uses/Calls:
Used by: Back Door Man

3.8 Customer Services

This category of services includes all information specific to the customer who initiates a session. All information related to identifying the customer, the issuing business of the customer, the customer's profile and all the customer's accounts are the component objects included in this category.

3.8.1 Customer-ID

An object that contains information and can answer questions about a customer's identity and associated information.

Responsibilities:
- Support query of customer ID and card information.
- Support update of customer ID and card information.

Knows:
- Customer primary ID (CIN & encrypted PIN/TPIN, Public Key Certificate)
- PIN/TPIN length
- Flag indicating ID validity: valid, invalid, unknown (used to tailor session flow).
- Card information (if a card was used)
  - ◊ Type of card (ATM/Credit, SmartCard)
  - ◊ Tracks present & track data
- The following items may be used provide a common place to store optional customer information that may be obtained from the customer or a external service provider during a session, so that it does not have to be requested more than once:
  - ◊ Name of customer
  - ◊ Mail address of customer
  - ◊ e-mail address of customer
  - ◊ Phone numbers (business, home)

Instances: One per session
Uses/Calls:
Used by: Welcome Mat (instantiated by)
Dialog Services
TXN Services

3.8.2 Issuer

This component represents the issuing business for the customer-id information that was used to start this session. This component is the rule authority for all general, issuer related, non Mini-App specific business rules.

Responsibilities:

- Support query of Issuer information.
- Support answering questions about general Issuer business rules.

Knows:

- Issuer of customer's identity (card issuer, or Public Key Certificate issuing authority).
- Issuer type (bank card, credit card, other 3rd party)
- Issuer's presentation rules (e.g. data, format, account number masking).
- Issuer's locale rules (e.g. collect call support, currency, product names).
- Issuer's server-ESP communication rules (e.g. profile message support).
- Languages supported.
- Navigation schemes supported.
- PIN length supported.
- Products supported.
- Services provided.
- Issuer country.
- When/how to authenticate customer (local validation of Public Key Certificate, immediate to Issuer, background to Issuer, delayed to first transaction).

Instances: One per session
Uses/Calls:
Used by: Welcome Mat (instantiated by)
Dialog Services
TXN Services
Customer Services

3.8.3 Customer Relationship

An object that contains information and can answer questions about a customer's relationship. This information includes: the accounts and products

owned by this customer, customer type (e.g. Citigold), preferences, and privileges.

Responsibilities:
- Support query of customer relationship information.
- Support update of customer relationship information.

Knows:
- Owner of the customer relationship (Issuer).
- Customer type (e.g. Citigold).
- List of accounts/products associated with the customer.
- List of possible payee accounts for this customer.
- Customer predefined transactions.
- Account category balances.
- Customer preferences (e.g. navigation style, standard transactions).
- Customer privileges or limitations.
- Transaction related dynamic counters (e.g. number of stock quotes allowed).
- Marketing messages (based on customer type).

Instances: Typically one per session
Uses/Calls:
Used by: Welcome Mat (instantiated by)
Dialog Services
TXN Services

Notes:
Some businesses (e.g. Mexico, Venezuela, and Brazil) can have multiple relationships per card. In the Top Level Navigation, the customer selects one of them as the primary relationship to use for the session. The Transfer application, however, can transfer between accounts in different relationships. It is not yet clear whether multiple relationships will need to be surfaced to all Mini-Apps, or whether only a few Mini-Apps will be able to deal with it. It is also not clear whether there will be one or multiple Customer Relationship components in this case. It could be that the list of accounts and products owned by a customer may be structured so that each item is associated with a different institution (account holder).

3.8.4 Account

An object that contains information and can answer questions about a particular account. There is one Account component per individual account. However the account details and rules will vary for different account types. This component may in turn use a TXN Executor for messages to external service providers to get information about the account.

Responsibilities:
- Support query of account information.
- Support update of account information.

Knows:
- Business owning the account
- Category of account
- Product type of account
- Account name (and font set required to display it)
- Account number
- Account details (currency types, balances, terms, etc.)
- Functional privileges and limitations (e.g. TJ supported for this account).
- Associated linked accounts (e.g. where to get money on overdraft).

Instances: One per account
Uses/Calls: TXN Executor (to get account info from external service providers)
Used by: Welcome Mat (instantiated by)
Dialog Services components
TXN Services components

Notes:
Accounts may be customer owned or payee accounts that can be the target of a transfer or bill payment.

3.9 Business Services

The components within the business service category provide formal mechanisms for dealing with business rules, language support and acquirer services.

3.9.1 Acquirer

An object that contains information and answers questions about the Acquirer. The Acquirer component represents the acquiring business for this session. The Acquirer component is the rule authority for business rules that are acquirer related, but not Mini-App specific. For rules that are acquirer related and Mini-App specific, separate rule authorities will be registered as part of the dynamic installation of a Mini-App.

Responsibilities:
- Support query of Acquirer information.
- Process certain specific rules associated with the Acquirer.

Knows:

- Acquiring business for this session.
- Acquirer's presentation rules.
- Acquirer's locale rules (e.g. collect call support, hours of operation rules).
- Acquirer's server-ESP rules (e.g. end of session MIS message supported).
- Country of Acquirer

Instances: One per session
Uses/Calls:
Used by: Dialog Services components
TXN Services components

3.9.2 Rule Broker

The Rule Broker formalizes a mechanism for dealing with business rules that up to this point in CAT applications has been ad hoc. The Rule Broker is the central registry for all business questions. Other components within a session, address named business questions to the Rule Broker. The Rule Broker routes the question to the rule authority or authorities that have registered for a rule. The design goal is to have a separable rule authority for each Mini-App specific business rule, so that new rules can be added independently without affecting the rest of the system. The Rule Broker mechanism also supports the concept of overrides. Overrides allow the dynamic registration of a new rule authority when changes to business rules are necessary.

Responsibilities:
- May either answer question directly or route question to another component (e.g. Account, Issuer, etc.)
- Responsible for interfacing into rules databases.

Knows:
- What component will answer each question.

Instances: One per session.
Uses/Calls:
Used by: Welcome Mat
Navigation Shell(s)
Mini-App Dialog(s)

Notes:
Refer to the Rule Broker appendix for more details.

3.9.3 Language Man

The Language Man provides the application with a facility to resolve the necessary text phrase needed in a particular context. The context includes the language selected by the customer and the type of device in use. It provides a repository of phrases which allows an application to be written in a language and device independent way.

From the application point of view all phrases are named. When an application needs to display a phrase, it asks the Language Man for the correct text for this phrase name given a specified language choice and the current presentation device type (provided by the Presentation Manager component). The capability exists to uses phrases with embedded variables. Thus, the application may supply additional parameters to be inserted into the phrase at the required point.

To resolve the request, Language Man uses a Phrase Repository to look up the correct version of a particular phrase. The Repository is segmented. There is a set of "global" phrases usable by all applications and a Mini-App Dialog specific set of phrases. Thus, given the ID of the requesting Mini-App Dialog, the repository specific to that Mini-App Dialog is searched first, and then, if the phrase is not found there, the global repository is searched. This allows a degree of independence in the creation of Mini-App Dialogs. No coordinated update to the Global repository is needed to release a new Mini-App Dialog and a Mini-App Dialog can override the Global phrase.

Language Man also provides APIs for the dynamic construction of phrases needed to deal with gender and plural issues encountered in some languages.

Responsibilities:

- For a specified language and device, it looks up the requested phrase in a Phrase Repository and returns it.

Instances: One per session.
Uses/Calls:
Used by: Welcome Mat
Navigation Shell(s)
Mini-App Dialog(s)

Notes:
The Language Man is backed by a set of development tools to create and maintain Phrase Repositories. These tools provide for:

- Creation and deletion of phrase IDs
- Mechanisms to add, change and delete phrase text in the repository.
- Multi-lingual text entry
- Specification of variable insertion points.

3.10 System Services

These components provide common services for all sessions within a server. The services include logging, event brokering, service registration, and cryptographic services. These components are:

Logger
Event Broker
Services Registry
Crypto Man

3.10.1 Logger

Writes and manages log files. This function is mostly handled by the NT log facility, but there may be a wrapper for it that provides some additional functionality.

Responsibilities:
- Adds standard headers to log entries.
- Writes them to a log entries to log.

Instances: One per server
Uses/Calls: Operating System services
Used by: All services

3.10.2 Event Broker

Provides a way for a business to do specialized processing of events (e.g. error conditions, transactions done, etc.) for the purpose of monitoring and acting upon activities in a server. Local business provided components can register with the Event Broker to receive specified events.

Responsibilities:
- Evaluate filtering rules associated with events and call the registered component as a result of a rule succeeding.

Instances: One per server
Uses/Calls: Any component that processes events.
Used/By: Any component that generates events.

Notes:
For example, the Event Broker could decide when to send notifications to a System Management system.

3.10.3 Services Registry

Responsible for registration of the Mini-Apps and Legacy App Bridges that are available.

Responsibilities:

- The Services Registry must work in the context of the procedures for software distribution, and cutover/fallback of releases in order to maintain a Registry of the Mini-Apps and Legacy Apps that are currently available.
- Provide information to the Navigation Shells about the Mini-Apps and Legacy Apps that are currently available.

Knows: The full set of Mini-Apps and Legacy Apps that are available.

Instances: One per server
Uses/Calls:
Used/By: Navigation Shell(s)

3.10.4 Crypto Man

Responsible for doing the cryptographic functions necessary to handle security. This component would manage any secret keys associated with external service providers, and would do authentication of Public Key Certificates.

Responsibilities:

- Hold security keys for each external service provider (may be multi-level keys per ESP). Keys may be shared secret or private key associated with a Public Key.
- Update keys.
- Use keys to generate message MAC and encrypt message.
- Encrypt (and re-encrypt) customer PIN/TPIN

Instances: One per server (probably per security mechanism)
Uses/Calls:
Used by: Front Door Man
Back Door Man

Notes:
In the current CAT, this is the Citishield device.

3.11 Peripheral Services

The components within the Peripheral Services category are responsible for handling application requests for peripheral device services and for managing the software components that handle such requests. Thus, these components are typically needed only for those products (e.g., CAT, CASST) that require custom high-level application interfaces to peripheral devices. The components in this category are:

Peripheral Device Manager
Peripheral Device Handler(s)

3.11.1 Peripheral Device Manager

Manages the components that interface with the connected peripheral devices.

Responsibilities (for device management):
- loads the peripheral device handlers for the connected devices during startup
- initializes the peripheral device handlers during startup
- notifies interested parties of changes in peripheral device availability (up, down)
- finalizes the peripheral device handlers during shutdown
- unloads the peripheral device handlers during shutdown

Responsibilities (for application services):
- coordinates usage of the peripheral devices by customers versus diagnostics
- serializes application requests to each peripheral device
- routes each application request to the appropriate peripheral device handler
- reports status of all connected peripheral devices upon request

Instances:	One per server
Interfaces:	MLI (for legacy components) COM (for new components)
Used by:	Welcome Mat, Mini-App Dialogs, TXN Executor
Uses:	Peripheral Devices, Logger, Event Broker

3.11.2 Peripheral Device Handler

Represents and controls a specific kind of connected peripheral hardware device.

There are several kinds of peripheral devices that may be connected to the service delivery platform for CAT and CASST products. Each peripheral device handler provides a generic device management interface and a specific service interface.

Peripheral device handlers may need to use specific subcomponents to interface with their associated peripheral hardware device.

Responsibilities (for device management)
- loads and activates needed subcomponents
- initializes the specific peripheral device hardware
- maintains persistent peripheral-specific management statistics
- reports peripheral-specific management statistics upon request
- reports peripheral device status upon request
- notifies interested parties of changes in peripheral device status
- recovers peripheral device hardware functionality when possible after a failure
- finalizes by releasing any needed system resources
- deactivates and unloads subcomponents

Responsibilities (for application services)
- tests the connected hardware device for correct operation
- normalizes the service interface so that similar devices share a common interface
- translates application requests into detailed hardware device requests

Note the specific nature of the services rendered depends on the specific device type.

Instances:	Many (one per connected peripheral device type)
Interfaces:	MLI (for legacy components) COM (for new components)
Used by:	Peripheral Device Manager
Uses:	Logger, Event Broker

Variations

The following table provides a representative list of the kinds of peripheral devices that might be connected to a CAT or CASST product platform. The first section lists those peripherals that are currently supported by the CAT and/or CASST platforms. The second section suggests some possibilities for future peripherals.

Peripheral Type	Action	Peripheral Data
Touch Screen	detects	Touch
Screen Display	displays	Screen
Form Printer	prints	Form
Card Reader	reads	Card
PIN Encrypter	encrypts	PIN
Envelope Depository	accepts	Envelope
Cash Dispenser	dispenses	Money
Speech Generator	generates	Speech
Sound Generator	generates	Complex Sound
Audio Generator	plays	Audio Clip
Video Player	plays	Video Clip
Proximity Detector	detects	Intrusion
Biometric Scanner	scans	Biometric Identifier

Examples

Representative examples of device management information include the following:

Peripheral Type	Status	Statistics
Card Reader	Up/Down	Cards Read
	Capture Bin Full	Bad Reads
		Cards Captured
		Long Timeouts
		Short Timeouts
Depositor	Up/Down	Envelopes Captured
	Ink Low	
	Bin Full	

Peripheral Integration Strategy

The architecture leverages Windows NT to support the integration of third party peripherals that provide standard NT hardware drivers. Thus, applications may be given direct access to peripheral device services through industry standard interfaces (e.g., scanners support TWAIN). On the other hand, some peripheral devices may still need custom high-level application interfaces (e.g., SmartCard).

The decision regarding whether to support an industry standard interface for applications within the architecture must be made for each new device. Some of the factors that influence a decision in favor of a custom interface include whether a standard interface exists for the device, the complexity of such an interface, and the complexity of the application interactions with the peripheral device. For example, while SmartCard uses industry standard interfaces for data communications (ISO 8825) and services (ISO 7816), the complexity of these interfaces and the peripheral device interactions influenced the decision to create custom high-level interfaces for the SmartCard applications.

4. Legacy Migration

The architecture supports an orderly migration of CAT functionality from implementation with AGS applications to implementation with Service Components on all platforms where AGS is used to deliver CAT look-and-feel functionality. In particular it will support the running of existing AGS Level 3 transaction applications from a Navigation Shell component. It will also support a complete session executed via an AGS Level 2 application and its associated Level 3 applications.

For example the following configuration will be possible for a CAT:

Function	Implementation
Local/In-Region Session Balance Inquiry Withdrawal Transfer Deposit Bill Payment	Session-Level Service Components Mini-App Dialog, etc. AGS Level 3 Mini-App Dialog, etc. AGS Level 3 Mini-App Dialog, etc.
ICC Session Balance Inquiry Withdrawal Transfer	Session-Level Service Components Mini-App Dialog, etc. Mini-App Dialog, etc. Mini-App Dialog, etc.
Off-us Session Balance Inquiry Withdrawal	AGS Level 2 AGS Level 3 AGS Level 3

4.1 Interaction Between CAT AGS Applications and Service Components

The basis for the interaction model is the manner in which session context is represented within the CAT AGS applications. AGS applications are executed within an instance of a TAFE process (the legacy runtime AGS Driver and

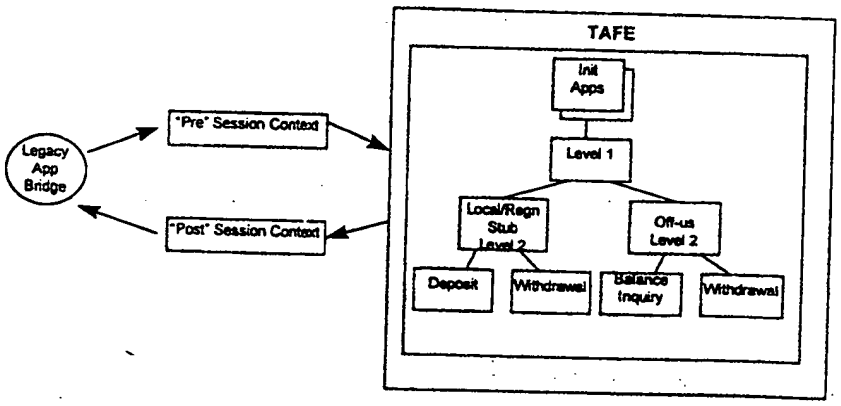
associated functionality) and share a single persistent global data store. At the time a CAT application is invoked session context is completely represented by two things: the current state of the persistent global data and the content of the Exit message TAFE passes to the application. If this context can be instantiated by alternate means then it is not necessary to perform the business/customer functionality normally performed by the AGS Level 1 and Level 2 applications before running a Level 3 transaction application.

At a high level the interaction model may be described by the following scenario #1:

- import pre transaction session context to TAFE
- invoke Level 3 application with Exit message
- return from Level 3 application with Exit message
- export post transaction session context from TAFE

For the case of a complete session performed in AGS the interaction is described by scenario #2:

- import pre language selection session context to TAFE
- invoke Level 2 application with Exit message
- return from Level 2 application with Exit message
- export post end-of-session context from TAFE

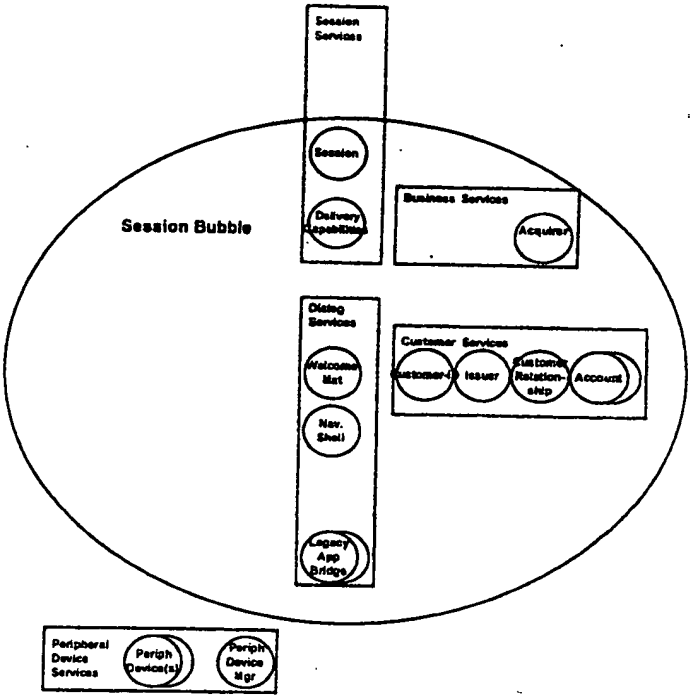


The vehicle for import and export will be ELF formatted messages that can be defined in an AGS Data Dictionary and received and sent by AGS applications. It is possible to define these messages to be composed of the persistent global variables and tables that comprise the necessary context such that no data manipulation is required in AGS after receipt of the import message or prior to sending the export message. The architecture does not specify the handling of these messages within TAFE or whether they are implemented as a single import message and a single export message per interaction. However, it is anticipated that scenario #1 can be supported without requiring modification to the subject Level 3 applications and scenario #2 can be supported without requiring modification to either Level 2 or Level 3 applications. It is also anticipated that there may be modifications to TAFE to support the interaction model.

The interaction model places two key responsibilities on the Service Components:

1) Collectively, the Service Components must capture and maintain sufficient session context information from which to derive the context representation required by the AGS application to be invoked. The required context will vary, in detail, by target AGS application set (USCAT, EuroCAT, AsiaCAT, LatinCAT, ICC, each particular off-us network) and whether the application being invoked is Level 2 or Level 3.
2) Each Legacy App Bridge component – whether representing an AGS session or an individual AGS transaction application – must be capable of constructing and interpreting ELF messages using the data name space (i.e. element id range) appropriate to the target AGS application.

The session context data required by legacy CAT AGS applications will be assembled from the properties and state of the Service Components in the figure below, or a subset, as required for the specific legacy functionality to be invoked.



The Legacy App Bridge component embodies the knowledge of the other components it must query and the specific properties it must access in order to assemble the session context that it must deliver to TAFE, and, likewise, the knowledge of the components and properties that must be updated by the modified session context at the completion of an AGS processed transaction or session.

CAT AGS Application Context Data

This section gives an overview of the context data used by CAT AGS applications. Two categories of context data may be distinguished: static data and session data. Static context data is instantiated at application startup and is comprised of business rules for both acquirer and issuer, text strings such as names and menu selections and CAT configuration information. Session context data includes card information, customer input information, customer profile data returned from the Issuer external service provider, device status, exception conditions and counters.

The following table shows examples classified by the CAT application that sets or modifies the data values, along with an indication of whether the data is imported or exported in the interaction model:

Context	Source Application	Examples	Import, Export
static data	initialization	issuer tables, acquirer tables and environment	—
	download	issuer tables	—
session data	Level 1	card info, start time, BIN table entry	import
	Level 2	language, no-record confirmation, transaction selection info, customer profile	import
	Level 3	encrypted PIC, counters, print record flags, printer availability	import, export
	Level 3	end transaction, end session, termination info	export

Note that in the interaction model static context data is not normally imported, rather it will continue to be completely instantiated as it has been in AGS. However, as the need arises, the import of context data can be used to override static data. This might be done when there are new or changed rules or display phrases that are applicable to the AGS application so that making the same change to both the rule database and a static AGS table (selm file), or to both a phrase dictionary and a static AGS table may be avoided.

Interaction with Devices and External Service Providers

CAT AGS Applications will continue to use the Logical Device Handler Message Level Interface to request device services, the AGS language PRESENT statements to display screens, the CAT/ESP Message Level Interface for interaction with the external service provider and the existing transaction logging message interface. No application changes will be required. TAFE will handle the necessary interactions with the Peripheral Device Manager, Presentation Manager, Back Door Man and Logger components.

Limitations

While it will be possible to run legacy applications in the new architecture environment, they will not support all of the interactions with the Navigation Shell

that the new applications can support, such as the ability to run more than one application at a time and switch between them.

4.2 Selectively Initiating Individual CAT AGS Applications

The precise mechanism for initiating a CAT AGS application is subject to design; the following elaboration of scenario #1 is a strawman meant to illustrate in more detail than the architecture specifies, how the interaction could occur. This scenario is for the case of invoking a Level 3 AGS application from a Legacy App Bridge component that represents a particular transaction type.

1) A pre-initialized TAFE (AGS Driver) process is associated with the Session Bubble. Within it a faceless Level 1 application waits on receipt of a start-of-session context message.
2) The Legacy App Bridge component for the customer selected transaction sends a start-of-session context message to TAFE including track 2 data (real or synthesized). This message does not contain data from a element id range specific to a card issuer.
3) The Level 1 application receives the message, updating session context in persistent global memory. Using the track 2 data, pre-initialized static tables and existing functionality the Level 1 application creates and sends the Exit message to invoke the Level 2 application appropriate to the card issuer.
4) In this scenario the Level 2 application is a faceless, special purpose replacement for the original Level 2 application. It is specific to the element id range of the issuer. It sends a request message for the remainder of the session context data. The request message is routed from TAFE to the Legacy App Bridge component.
5) The Legacy App Bridge queries other Service components in order to construct and return a response message containing the remainder of the session context, including data in the element id range specific to the Level 2 application that sent the request.
6) The Level 2 application receives the message, updating the session context in persistent global memory. Using the transaction type code, language code and application state code received in the context data, together with existing functionality, the Level 2 application creates and sends the Exit message to invoke the Level 3 application appropriate to the transaction type.
7) The Level 3 application processes the transaction. It presents screens, sends and receives external service provider messages, device

messages and logging messages, and updates session context in persistent global memory. Upon completion it sends an Exit message to return to the Level 2 application.

8) The Level 2 application sends a message containing the updated post transaction session context which TAFE routes to the Legacy App Bridge component. The Level 2 application also sends an Exit message to return to the Level 1 application.

9) The Level 1 application waits on receipt of another start-of-session context message.

10) The Legacy App Bridge component receives the post-transaction session context and processes it causing the session context to be updated in the other appropriate Service components.

In the above scenario the Level 1 and Level 2 applications perform no customer or business functionality. Their role is limited to receiving and returning context data and invoking the appropriate lower level application.

5. Tools and Languages

The selection of tools and programming languages will be a follow on task to the specification of the delivery system architecture. This section is intended to give some direction to that process.

No new programming languages will be developed or defined to implement this application architecture. Existing languages can and will be used.

The goal of the delivery system architecture is to be language neutral. The applications can be written in any language which supports the object model used to specify the architecture. This means that different components may be implemented in different languages. It also means that we may migrate to a different (presumably better) language over time. In the near term, both Visual Basic Script and C++ are candidates for implementation languages. Longer term, Java will probably be a candidate for some components of the architecture, but, it is not yet mature enough for immediate use. By following a language independent object model, we can switch languages when appropriate.

The selection of an Integrated Development Environment (IDE) will be an important part of the follow-on tools and languages selection task. Some important criteria should be considered:

- The IDE should have support for multi-user shared development.
- Integration with a configuration management capability.
- It should support a tool "plug in" capability to allow us to add tools which are unique to our own development needs.

Additional tools which we will need to purchase or develop to "plug into" the development environment include:

- Configuration tools to allow for the maintenance of system configuration information.
- Test tools including Host and Device emulators.
- Software Distribution tool to standardize the method by which software updates are distributed.
- System Management and Logging Tools

- Security Protocols
- Middleware for Distributed Object support and Legacy System Interfaces.
- Template Development Tools for both canonical and device specific templates
- Rules Data Base Editor
- Services Registry Maintenance Tools
- Language Man Phrase Repository Editor

Ideally, the same IDE should support all of the selected targeted languages. This will minimize retraining and allow re-use "plug in" of tools across development languages.

Appendix A - Rule Broker (explained)

A key element of the architecture is the separation of individually-installable Business Rules from the code embodied in the transaction-specific components. Application components needing answers to Rule questions ask the Rule Broker, without knowing any details about how the rules are encoded and answered. The Rule Broker routes the question to the appropriate component which can supply an answer. Components which supply Rule answers may be installed independently of components which ask the Rule questions. In addition, any data used by a Rule "answerer" may be installed or replaced independently from components which use that data to determine answers to Rule questions.

A Business Rule is a statement of policy driven by business or regulatory needs, which defines context-specific behavior of the applications. Business Rules in the architecture are discrete items which may be modified independently from other application components.

The following sections summarize the major components which support the "Rule Broker" function in the Application Architecture.

A.1 Rule Broker

A single entity which components of the application may access to obtain answers to the Business Rules questions which affect application processing.

The Rule Broker provides a mechanism for Rule Authorities to register themselves as answerers for particular rule questions. When application components query the Rule Broker for a particular Rule, the Rule Broker routes the query to the appropriate Rule Authority or to the Rule Engine. The Rule Broker itself is not aware of the actual semantics of any of the rules.

Responsibilities:

- Receives Rule Registration requests.
- Registers Rule in a Rule Registry.
- Receives Rule queries and routes them to the registered provider for that rule.

Instances:

- One per session.

Uses/Calls:

- Rule Engine
- Rule Authorities

Used by:
- Mini-App Dialogs
- TXN Executors
- Presentation Manager
- Navigation Shell
- Welcome Mat
- Legacy App Bridge

Notes:
- The presence of a Rule Broker does not preclude individual components in the system from being direct (non-brokered) "answerers" of questions, when appropriate.

A.1 Rule Authority

A component which can answer Rule questions.

Components in the architecture act in the role of Rule Authority if they register themselves with the Rule Broker as the answerer of a named Rule. For example, the Issuer, Acquirer, and Delivery Capability components may be Rule Authorities.

Responsibilities:
- Register Rule(s) with the Rule Broker.
- Provide answers for the Rule(s) this component is registered for.

Instances:
- One to many Rule Authorities for each registered Rule.

Uses/Calls:
- Rule Broker
- Rule Database

Used by:
- Rule Broker

Notes:
- Rule Authorities may access separately installable data to answer Rule questions. This data is separate from the Rule "Registry" information used by the Rule Broker and the Rule Engine.

A.3 Rule Engine

A general Rule interpreter.

The Rule Engine can answer a Rule query based on parameters passed in the query, and some interpretable Rule data in the Rule Database. Unlike Rule Authorities, the Rule Engine has no specific knowledge of Rules or applications.

Responsibilities:
- Determine answers for Rules.

Instances:
One per session.

Uses/Calls:
- Rule Registry

Used by:
- Rule Broker

A.4 Mechanism

The following points summarize the mechanism to be used to resolve Business Rules questions:

- Each Rule registered with the Rule Broker will have a unique Name which includes a version identifier.
- The Name will be passed separately from other parameters in a Rule query.
- All Rule query parameters aside from the Name will be passed in a self-defining way. (e.g., a Rule query may contain a name, type, and value for each parameter).
- Each Rule registered with the Rule Broker will exist as an independent record in a Rule Registry.
- A Rule in the Rule Registry will be defined as either:
    a) Data (e.g., an encoded string) which can be interpreted by a general Rules Engine, or
    b) A Rule Authority which is registered to answer a Rule. It will be possible for a component's registration of a Rule to override a previous component's registration for that same Rule.
- Each registered Rule will define the expected type of parameters to be passed.
- Rules can be dynamically added to the Rule Registry independently of all other Rules.

- The Rule Broker will route a Rule query either to the general Rules Engine, or to a sequence of Rule Authorities (until an answer is obtained or no more authorities are available).
- The Rule Broker routes queries based only on the Rule Name.
- The Rule Broker does not validate the parameter list. It is the responsibility of either the Rule Engine or the Rule Authority(ies) to validate the parameters.

A.5 Protocol

The following points summarize the protocol to be used by the Rule Broker and by components querying the Rule Broker:

- Any component querying the Rule Broker must be prepared to handle the case of "no answer" (e.g., there is no such Rule registered, the component registered to answer the Rule cannot answer, etc.) gracefully.
- The Rule Broker must return a specific "no-answer" answer to a requester when no answer is available.
- The Rules Engine and all Rule Authorities must dynamically check the parameter list and return the appropriate "no answer" if there is a discrepancy between expected and received parameters.

A.6 Assumptions

The following assumptions are made regarding the Rule Broker and related components:

- There is a well known set of system and session components which provide context for rule questions and which are available to the Rules Engine or any Rule Authority. Attributes which are available from this context need not be passed as parameters.

A.7 Examples

The following are examples of possible uses of the Rule Broker. Note that these are only hypothetical examples that must assume some implementation details which will change when the detailed application framework design is done.

Example 1, Dispense Amounts

A set of complex rules, spanning multiple configuration tables, is used in the AGS implementation when choosing what Dispense Amounts are displayed on

selection buttons to a customer withdrawing cash at a CAT. The existing "Withdraw Cash" application is tightly coupled to the structure of these tables.

The Acquirer component might register as the Rule Authority for the "WhatDispenseAmouts?" question:

Name =	"WhatDispenseAmounts?"
Input Parameters =	ProductType (product being withdrawn from)
	Currency
Output Parameters =	ResultCode
	Variable length list of amounts

Note that some of the session data needed to answer the question (e.g. Card Type, Level of Service) is available from known session components, and so is not passed as input.

The Acquirer object, in processing the request, may query whatever database contains specific rules for dispense amounts (the equivalent of the current CAT configuration tables), and ask the Peripheral Device Manager to find out what denominations are actually available.

Example 2, Maximum PIC Tries?

The Rule "MaxPicRetries?", to be processed by the Rule Engine is registered in the Rule Database:

Name =	"MaxPICRetries?"
Input Parameters =	None
Output Parameters =	ResultCode
	MaxPICRetries
Rule Data =	Some interpretable data which indicates that a "business options" table should be searched for the MaxPICRetries value matching the session values of Issuer and CardType.

Note that all of the session data needed to answer the question (e.g. Issuer, Card Type) is available from known session components, and so no specific input parameters are needed.

The Rule Engine searches the specified table for a match on the session Issuer and CardType, and returns the value of MaxPICTries for that match.

CITICORP

NT Self-Service Delivery System Architecture

Architecture Task Force

DD/TTI

Our Process

Step 1. Define a conceptual architecture
(the major components and their relationship)

Step 2. Select tools/technologies to implement the components

Step 3. Detailed analysis of the components.("The What")

- ✧ Interface contracts
- ✧ Component functions
- ✧ Component hierarchy

Step 4. Detailed design of components ("The How")

- ✧ Refine interfaces
- ✧ Detailed internal design of components

Step 5. Implementation of the components

Architecture Task Force Members

- Jim Zeanah
- Chuck Abbott
- Nik Boyd
- Albert Cohen
- Jim Cook
- Michael Grandcolas
- Sikun Lan
- Bonnie Lindsley
- Grigor Markarian
- Les Moss

Objectives

- Common application base for customer activated applications running on
  - Remote access devices
  - CAT/CASST platform
  - Branch and CSR staff platform
- Convergence to a base set of re-usable global application components
  - Allow re-assembly to form new applications
  - Complemented by components from local business

Objectives (con't)

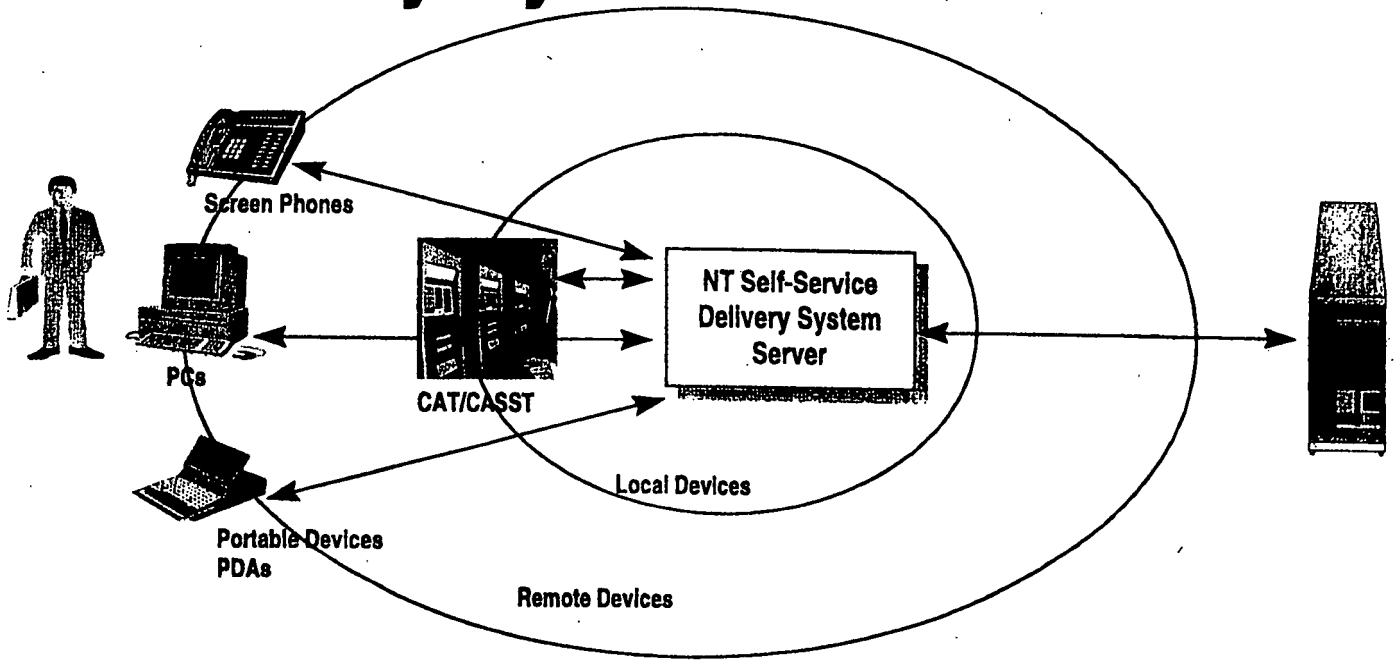
- Provide state of the art user interfaces
  - Integration of industry available multi-media
  - Support customization
    - specific devices
    - languages
    - countries
  - Multiple co-existing Navigation paradigms
    - Forms
    - Question and Answer
    - CAT Look and Feel
  - Access to concurrently active applications

Objectives (con't)

- Improve development and maintenance cycle time
  - "Re-use" of warehoused components
  - Use of industry standards for component interfaces
  - "Plug and Play" application components
    - Dynamic application registration
    - Automatic insertion in the top level navigation menu
  - "Modular" application components
    - "Independent" certification and warehousing

- Provide smooth gradual migration from legacy applications
  - Harmonious co-existence of new applications with existing legacy AGS applications

Scope of NT Self-Service Delivery System Architecture



- Self Service terminals (CAT and CASST)
- Remote Customer devices (PC, Screen Phone or PDA)

Our Approach

- **Our Focus**
  - Provide a conceptual application architecture
  - Define high level components of architecture
  - Define interactions among components
- **Leverage Industry Standards**
- **Leverage Domain Specific Knowledge**
  - Global Home Services Delivery System used as a base model

Business Rule

- A business rule is a statement policy driven by business or regulatory needs, which defines context-specific behavior of the applications.

- Business rules in the architecture are discrete items which may be modified independently from other application components.

Business Rules Examples

1. Choosing Dispense Amounts to display
2. Maximum PIC retries
3. Assignment of Product Types to Summary Categories
4. Assignment of Product Types to Product Categories
5. Number of Account Digits on Print Record

etc.

Rules of Thumb for Business Rules

On average there are:

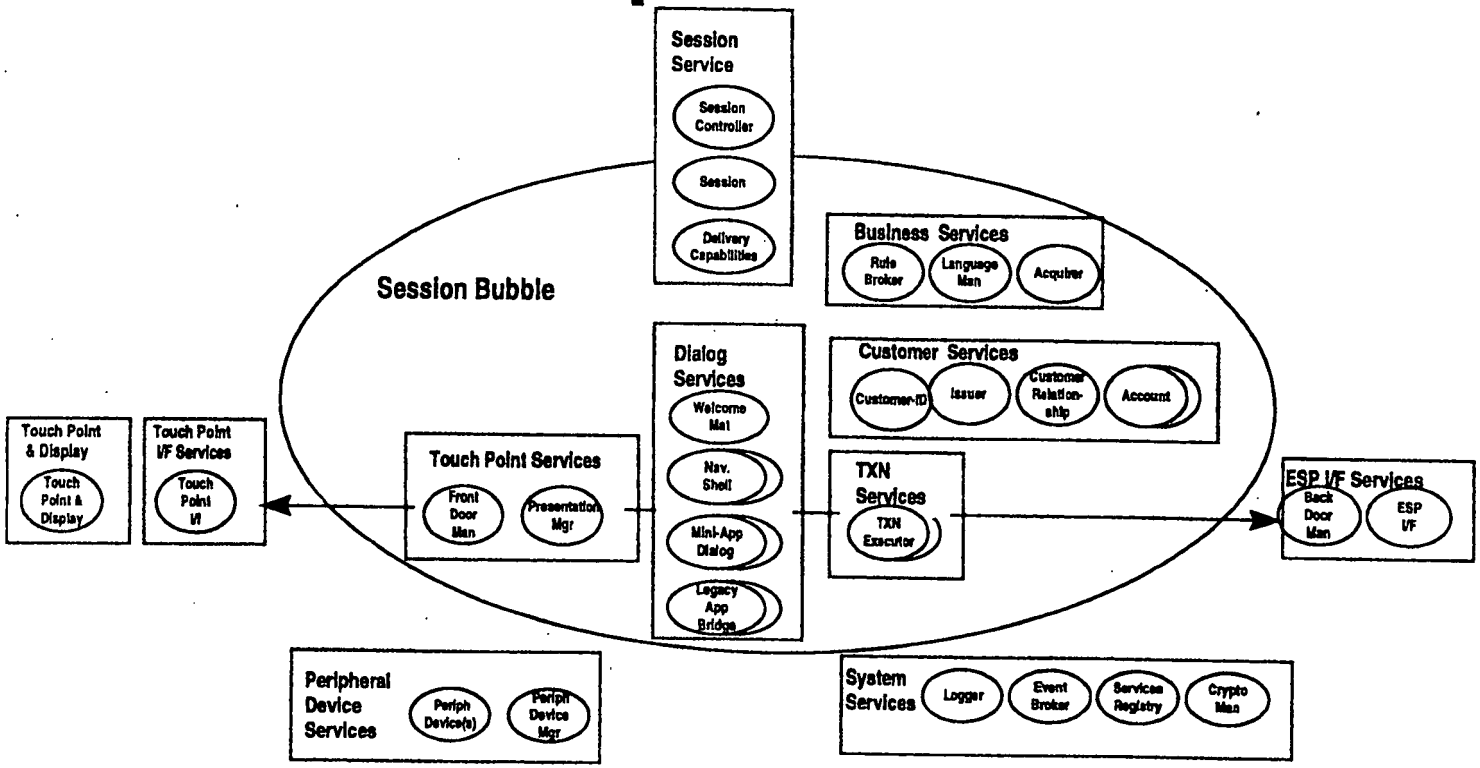
1. 50-100 runtime business rules per region
2. Most are Issuer Rules
3. Fewer Acquirer Rules (on the order of 12 or so)
4. 50 of the rules apply to all mini-apps
5. Fewer rules per mini-app (on the order of 0-12)
6. 12 mini-apps per regional set

Encapsulation

1. Hiding the details of the implementation.
2. Separating "what" from "how".
3. There are many possible "how's" for implementing the details of any business rules.
   a. In code.
   b. In a configuration table on the CAT
   c. In a configuration table on the External Service Provider
   d. In a configuration table cached on the CAT
   e. In a general purpose rules engine

   etc.

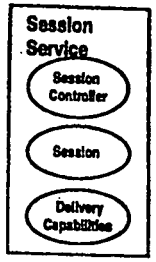
Conceptual Model



- We have modeled the architecture as a set of service components

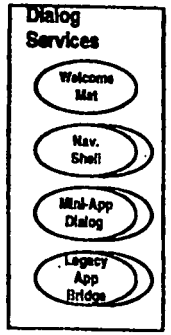
Major Service Components

Session Services



- **Session Controller Component**
  - Manages all sessions in the server
  - Starts a secure session bubble on behalf of the customer
- **Session Component**
  - Instantiates initial session resources
  - Manages resources associated with a session
- **Delivery Capabilities Component**
  - Contains all information about the capability of a Touch Point

Dialog Services



- ✧ Content and interaction with the customer
- ✧ Initiate transactions on customer's behalf

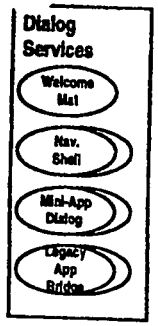
◆ **Welcome Mat Component**

- ✧ Display "Welcome" page to customer
- ✧ Collect preferences (Language, Navigation Style)
- ✧ Identify customer (e.g. Card dip on CAT)
- ✧ Identify "issuer" and authenticate customer
- ✧ Gather customer relationships

◆ **Navigation Shell Component**

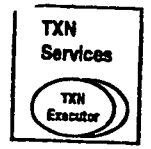
- ✧ Identify choice of Mini-Apps available to customer
- ✧ Assign a viewable frame space for each Mini-App
- ✧ Allow switching between concurrently active Mini-Apps

Dialog Services (con't)



- Mini-App Dialog Component
  - Manages dialog with the customer for a business function (e.g. Transfer, Bill Payment)
  - Supports a single dialog style (e.g. Form, Q & A)
  - Responsible for content and control flow of a business function
  - Interface with TXN Executor to perform transaction
- Legacy App Bridge Component
  - Enabling legacy app(s) to operate in new framework
  - Normalize data flow between legacy app(s) and new service components
  - Potentially can enable decomposition of individual AGS Level 3 applications

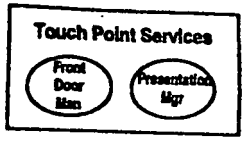
Transaction Services



- Interface with External Service Providers
- Interface with Peripheral Device services

- **TXN Executor Component**
  - Validate parameters obtained from Mini-App(s)
  - Perform necessary message formatting for ESP
  - Orchestrate complex transaction with multiple ESPs
  - Operate Peripheral Devices to carry out function (e.g. dispense cash)
  - Handle error recovery (may entail reversals)

Touch Point Services

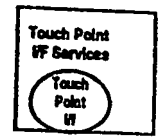


- Delivery vehicle specific presentation layout
- Enforce security with touch point devices

- **Presentation Manager Component**
  - Mapping canonical representation of information to delivery vehicle specific presentation format
  - Delivery Vehicle specific templates enforce consistent layout style

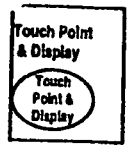
- **Front Door Man Component**
  - Add secure session token to outgoing messages
  - Verify session token on incoming messages

Touch Point Interface Services



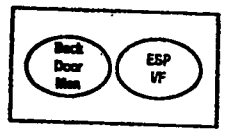
- **Touch Point Interface Component**
  - Manages the link/session level protocols with a Touch Point
  - On initial contact instruct Session Controller to initiate a session
  - Normalizes and routes messages from Touch point to appropriate Front Door Man
  - For Internet sessions, vendor provided Web Servers implement this service

Touch Point & Display



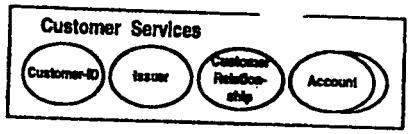
- **Touch Point & Display Component**
  - Actual customer touch point display and input facility
  - Manages the link/session level protocols with an application server on the touch point device
  - Decodes the server interface protocol and outputs a page to the local screen
  - Acquire user input, encode in server interface protocol and send it to the Touch Point Interface
  - For Internet sessions this is a Web Browser

External Service Provider Interface Services



- **Back Door Man Component**
  - Multiplex messages from TXN Executor(s) in several sessions to a single ESP
  - Message sequencing support
  - Handle response routing
  - Message security with ESP (e.g. MAC, encryption)
  - Perform automatic retries if required
  - Alternate routing to backup ESP upon detection of failure
  - Routing of unsolicited incoming messages
- **External Service Provider Interface Component**
  - Protocol support for a specific type ESP interface (e.g. X.25, SNA)

Customer Services

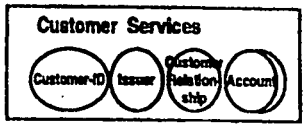


- Information specific to the customer initiating session

- **Customer ID Component**
  - Query of customer ID and card information
  - Update of customer ID and card information

- **Issuer Component**
  - Represents issuing business for the current customer
  - Query of Issuer information
  - Update of Issuer information

Customer Services (con't)



- **Customer Relationship Component**
  - Represent customer's relationship
  - Query of Relationship information
  - Update of Relationship information

- **Account Component**
  - One account object per individual account
  - Can interface with TXN Executor to gather information about account from ESP
  - Query of account information
  - Update of account information

Business Services

- Provide formal mechanisms for dealing with business rules, language support and acquirer services

◆ Acquirer Component

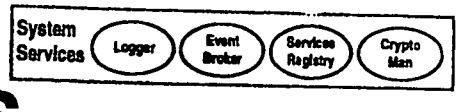
- Represents acquiring business for this session
- Query of Acquirer information
- Process specific rules associated with the Acquirer

Business Services (Con't)

- **Rule Broker Component**
  - Formalize a mechanism for dealing with business rules
  - Central registry for all business questions
  - Route questions to rule authorities
  - Each Mini-App specific business rule may have separable rule authority

Business Services (Con't)

- **Language Man Component**
  - Look up the requested phrase in a Phrase Repository based on specific language selection and presentation device
  - Phrase Repository segmented based on individual "Mini-Apps" and "Global" phrases
  - Support insertion of variables in phrases



System Services

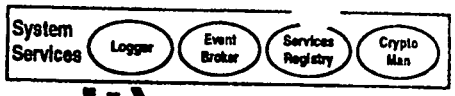
- ✧ Provide common services for all sessions

◆ Logger Component

- ✧ Write and manage log files in a standard format

◆ Event Broker Component

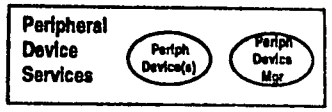
- ✧ Evaluate filtering rules associated with events and call the registered component
- ✧ Extend application processing by local business based on specialized processing of events (e.g. error conditions, transaction done)



System Services (con't)

- **Services Registry Component**
  - Repository for dynamic registration of "Mini-Apps" and "Legacy App Bridges" available in the system
- **Crypto Man Component**
  - Perform cryptographic functions necessary to handle security
    - Manage secret keys associated with external service providers
    - Perform authentication of Public Key Certificates

Peripheral Services



- Peripheral Device Manager Component
  - Manages the components that interface with the connected peripheral hardware devices
- Peripheral Device Handler Component
  - Represents and controls a specific type of connected peripheral hardware device

Rendering Model

- Basic Model for Rendering is "indirect"
- App Stream is HTML
- Dialog Services Components responsible for Content
- Presentation Manager Component responsible for Style
- Delivery Vehicle Templates define Style
- Separation of Content from Style
- Abstraction of Objects or Tokens
- Use of "ActiveX" Controls

Rendering Model (con't)

- Presentation Model - Templates, Frame Sets, Frames
- Accommodating Legacy Apps within a Frame
- Links and Input Data
- Basic App Stream Interface is Easy to Produce
- Advanced Object References in the App Stream
- Delivery Vehicle Specific Dialogs

Mini-App Packaging Model

1. The Mini-App Component.
2. Any TXN Executor Components needed.
3. Rule Entries to be registered with Rule Broker.
4. A Rules Engine File per Rule where appropriate.
5. Rules Database per Rule used by Rule Authorities.
6. Language file for Mini-App specific language phrases.

Mini-App Packaging Model (con't)

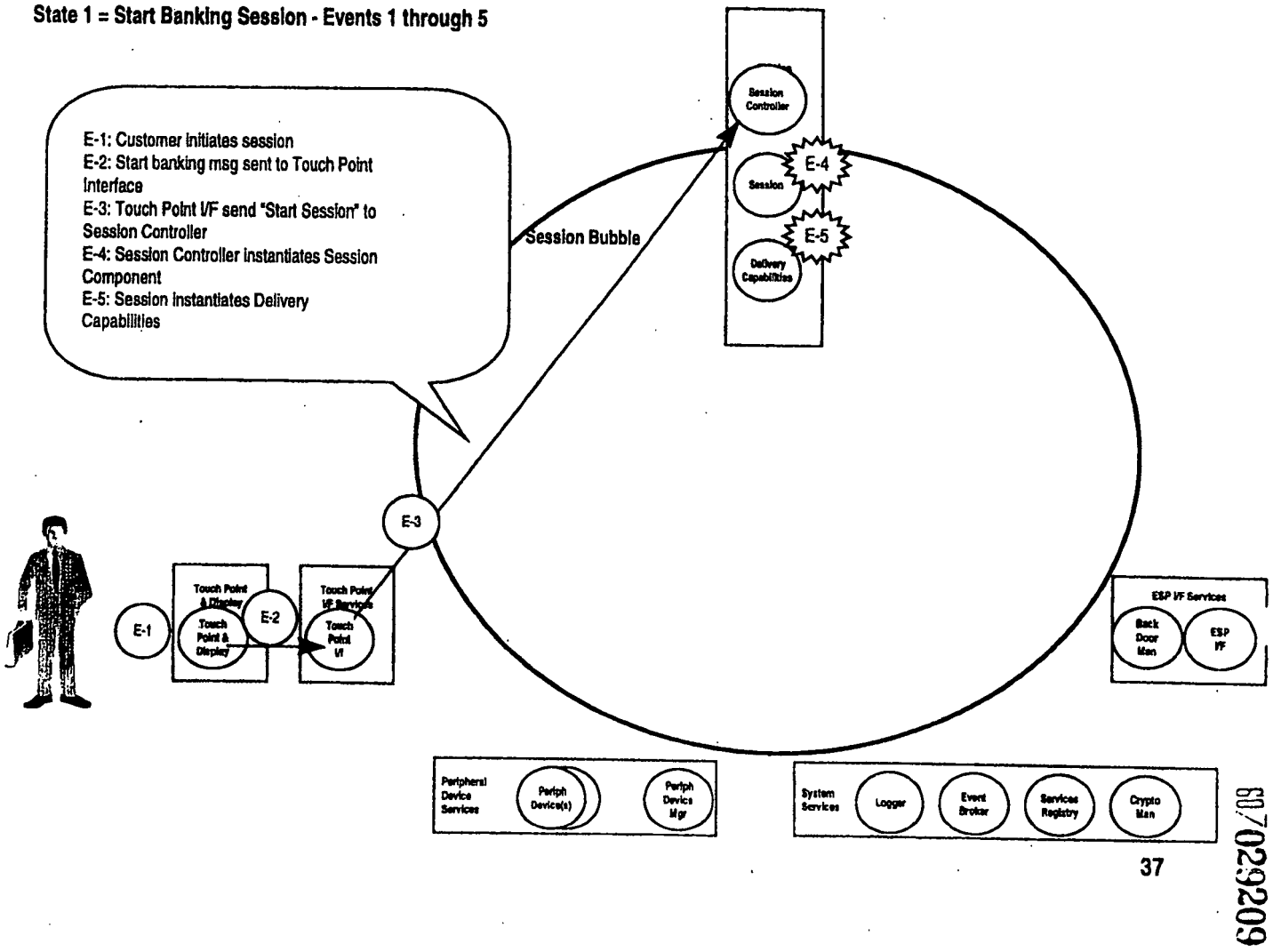
- Elements of a Mini-App Package include:
  - Executable (.exe) for the Mini-App component, including the TXN Executor component (DLL or OLE object)
  - RULE file including all new Rule Entries to be registered with the Rule Broker
  - Where appropriate, a Rules Engine file per rule, for any rules that can be interpreted by the general purpose Rules Engine

Mini-App Packaging Model (con't)

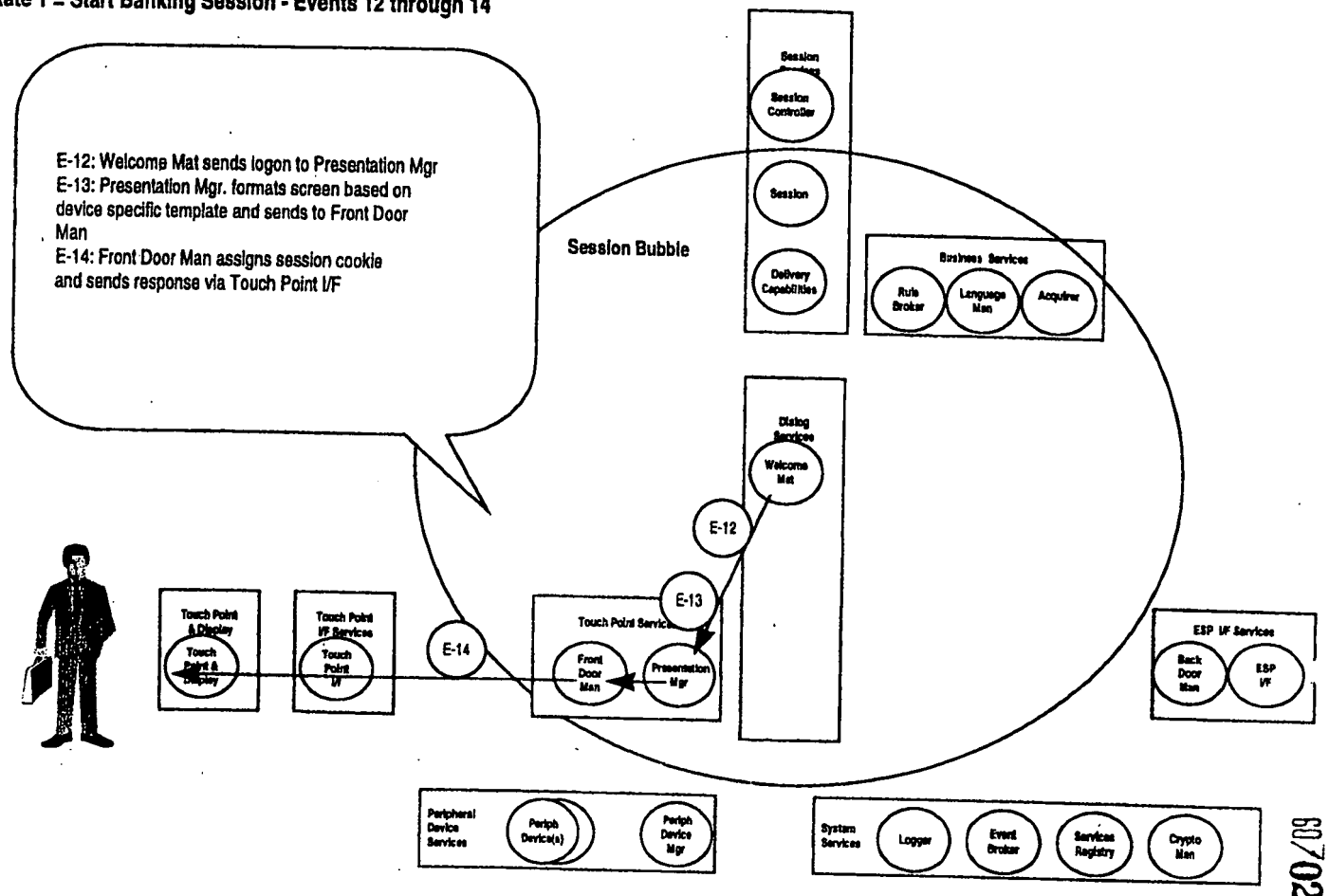
- **Elements of a Mini-App Package include:**
  - **Where appropriate, a Rule Database file per rule, that supplies any needed data to support Mini-App specific rule authorities**
  - **Language file including all Mini-App specific language phrases needed**
  - **Where appropriate, a Template file containing all Mini-App specific templates**

Conceptualized Session Walk-Thru

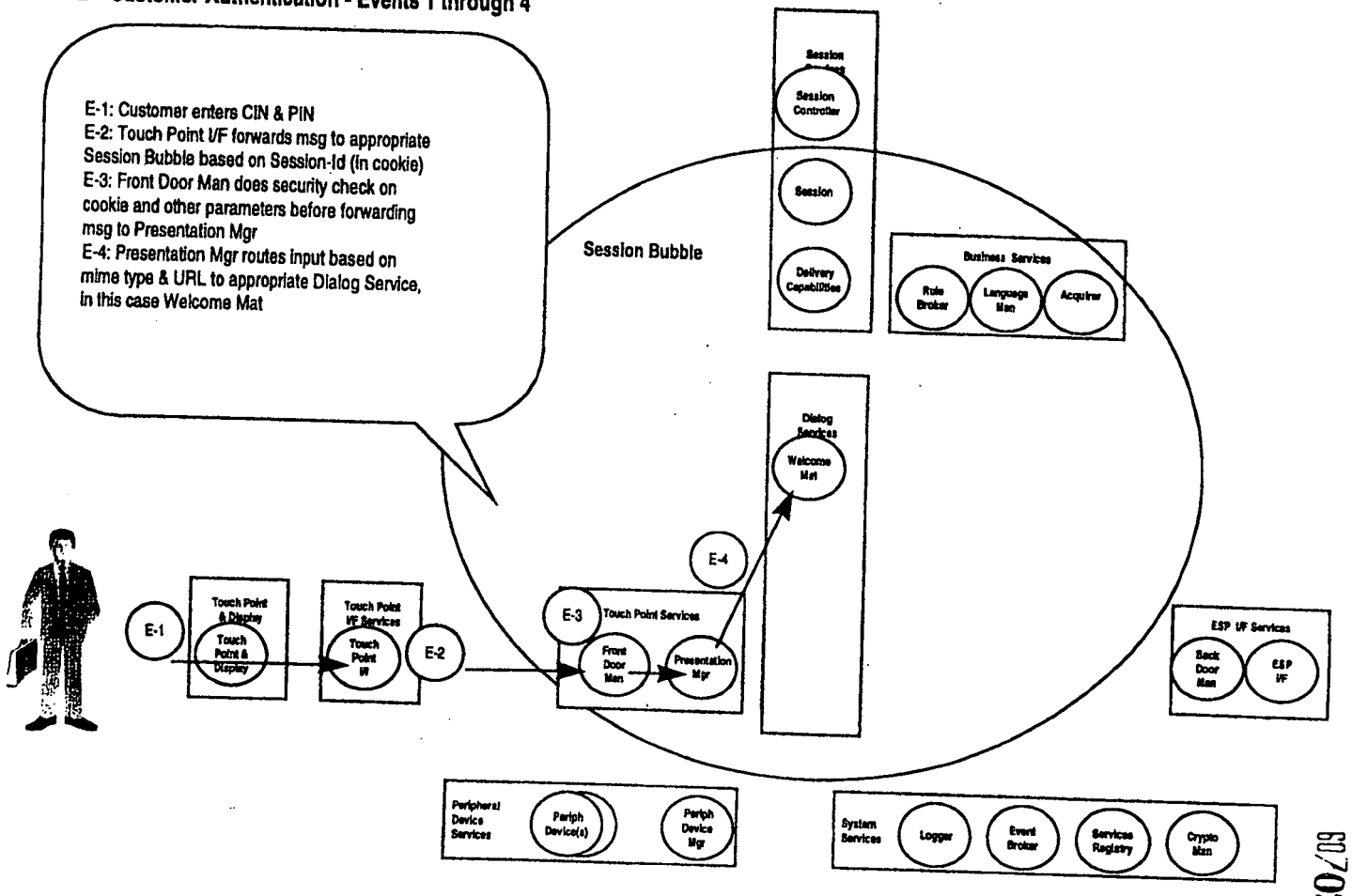
State 1 = Start Banking Session - Events 1 through 5



State 1 = Start Banking Session - Events 12 through 14

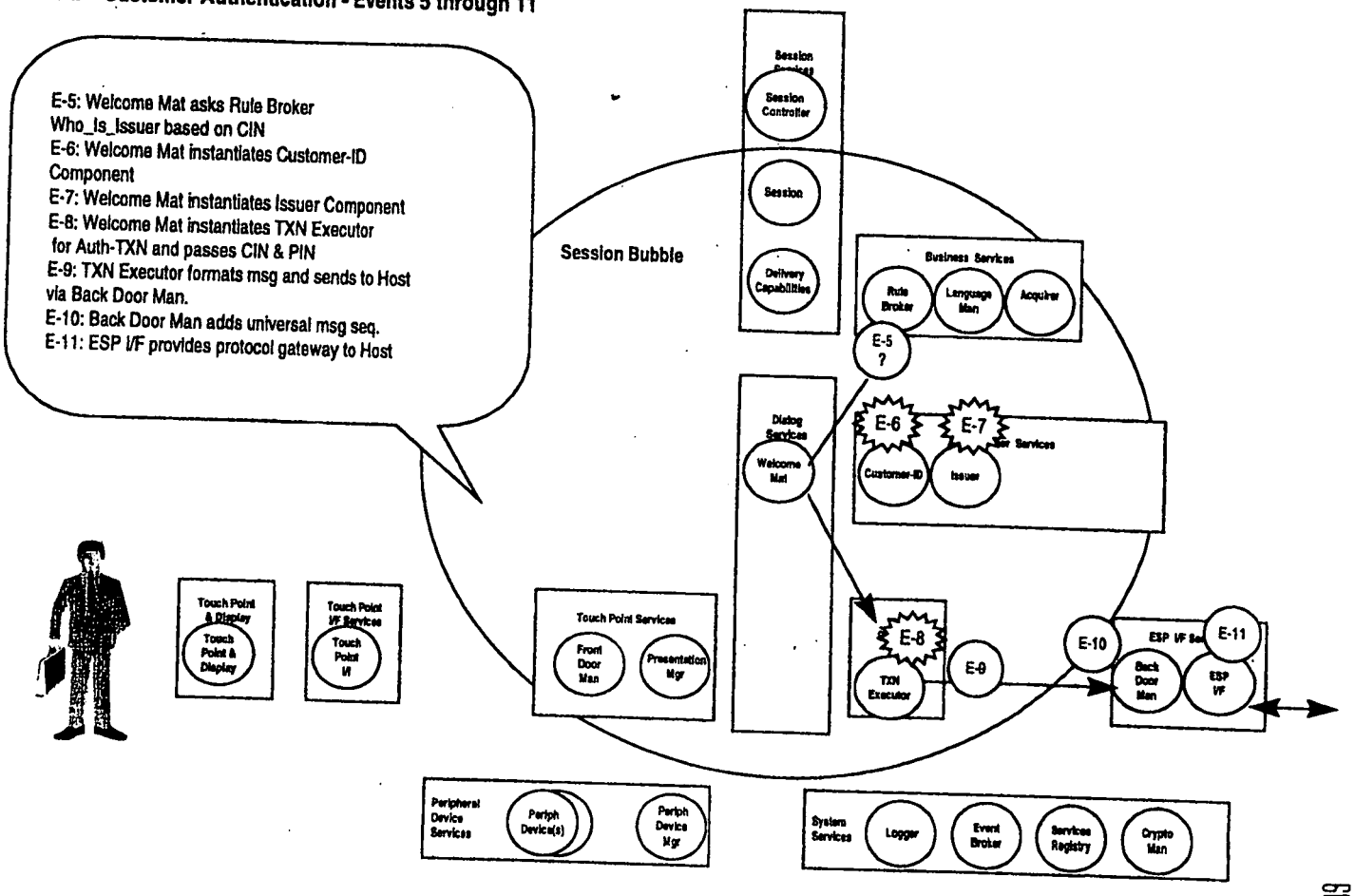


State 2 = Customer Authentication - Events 1 through 4



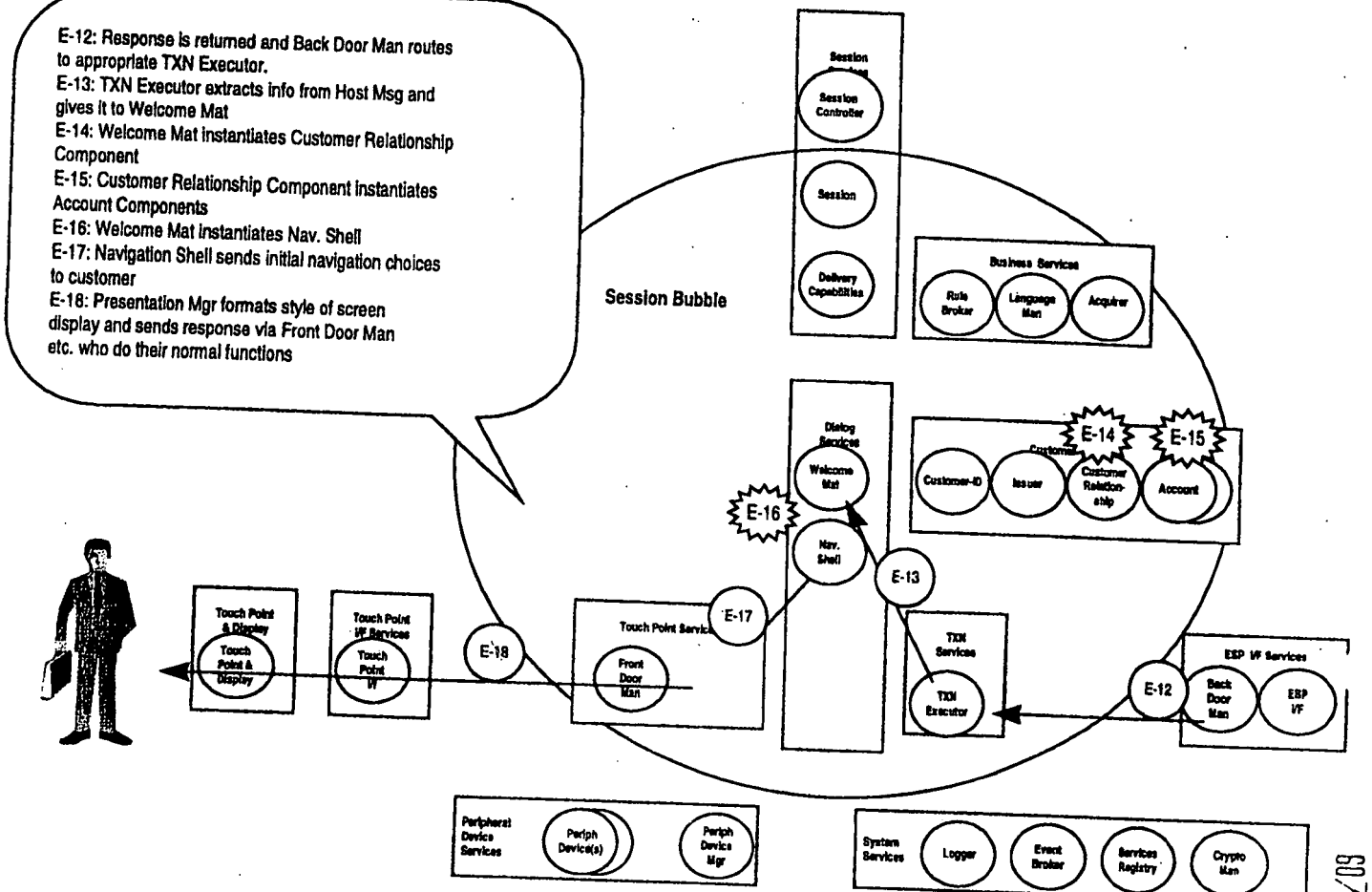
State 2 = Customer Authentication - Events 5 through 11

E-5: Welcome Mat asks Rule Broker Who_Is_Issuer based on CIN
E-6: Welcome Mat instantiates Customer-ID Component
E-7: Welcome Mat instantiates Issuer Component
E-8: Welcome Mat instantiates TXN Executor for Auth-TXN and passes CIN & PIN
E-9: TXN Executor formats msg and sends to Host via Back Door Man.
E-10: Back Door Man adds universal msg seq.
E-11: ESP I/F provides protocol gateway to Host

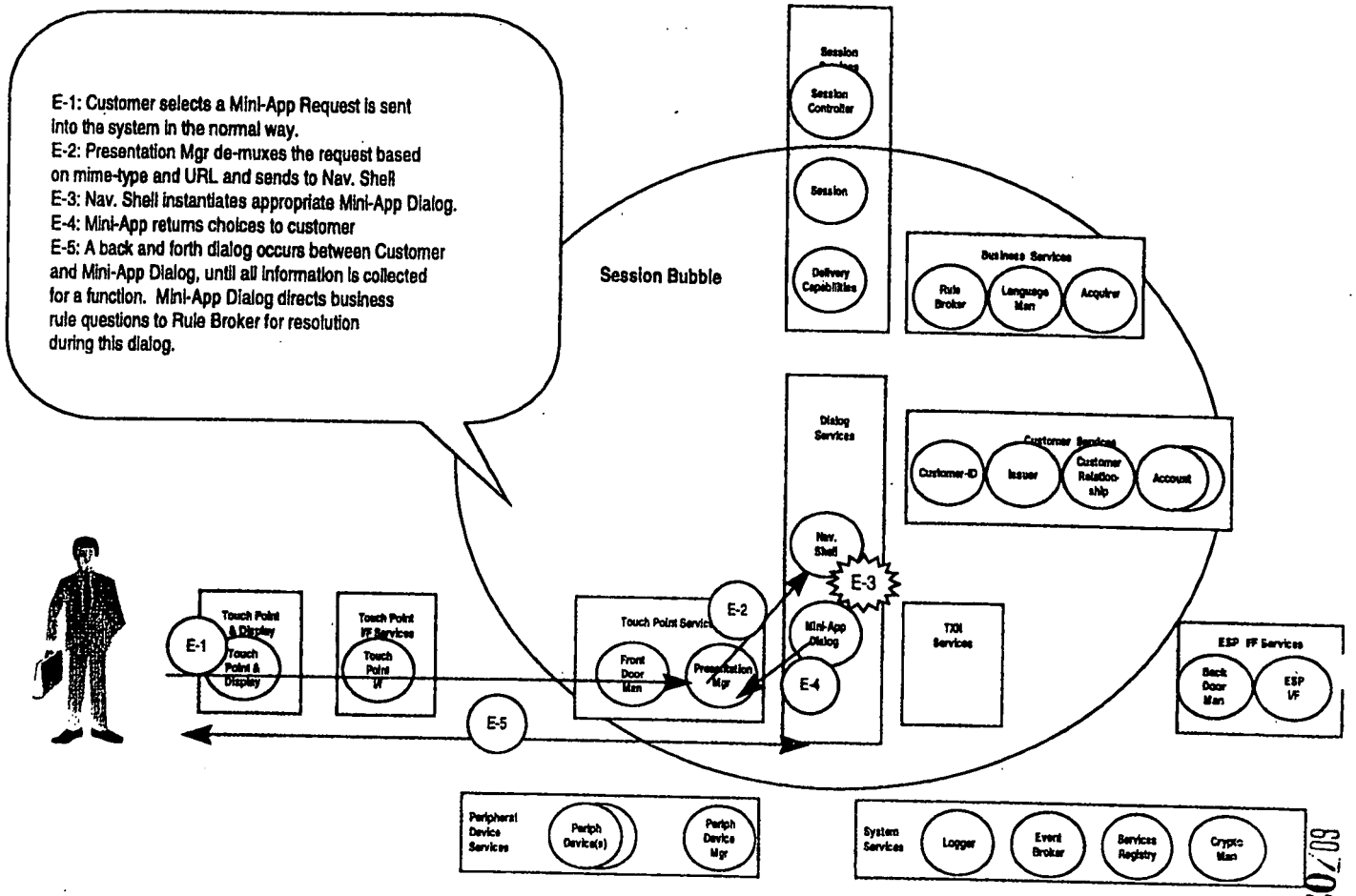


State 2 = Customer Authentication - Events 12 through 18

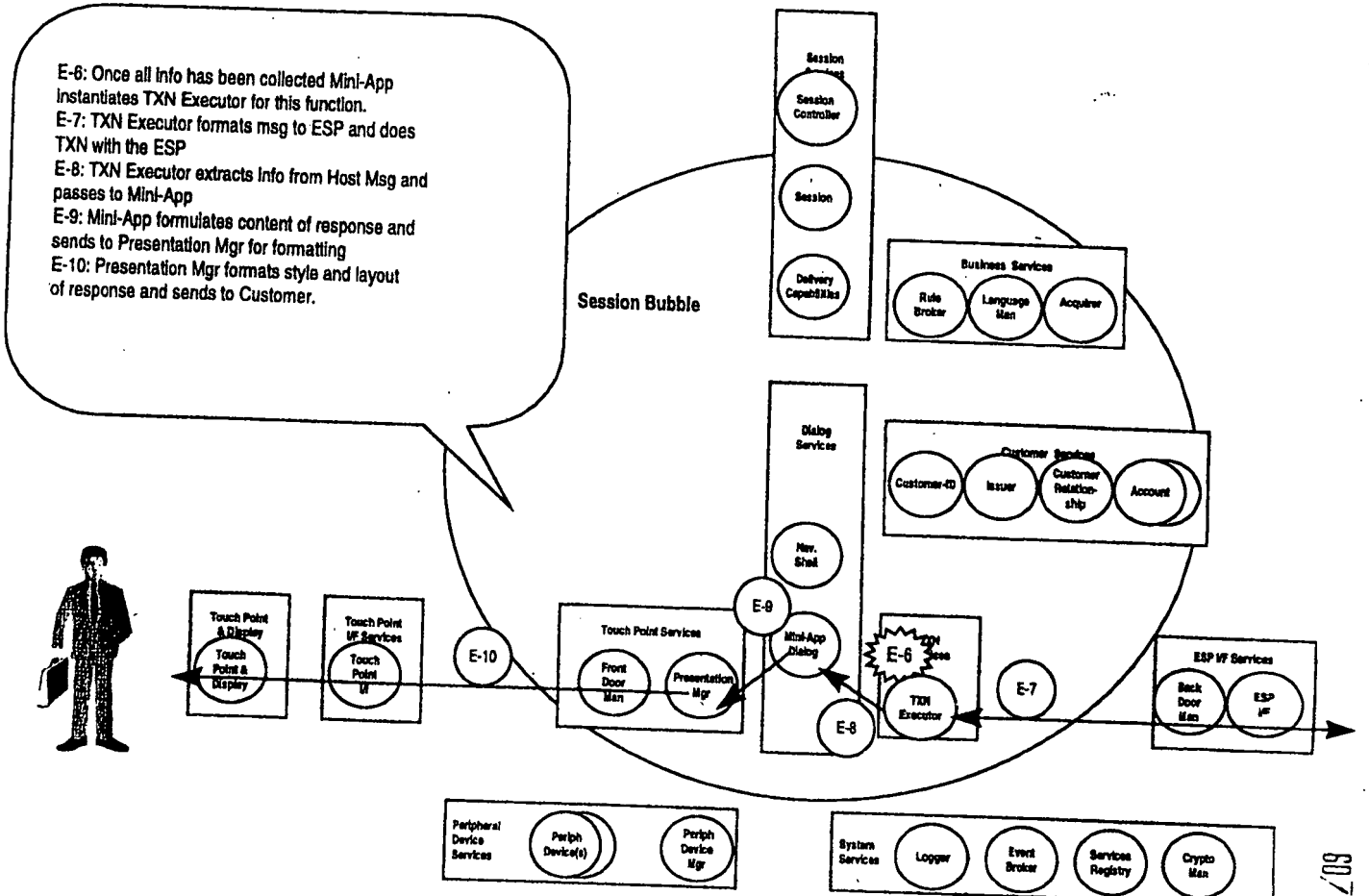
E-12: Response is returned and Back Door Man routes to appropriate TXN Executor.
E-13: TXN Executor extracts info from Host Msg and gives it to Welcome Mat
E-14: Welcome Mat instantiates Customer Relationship Component
E-15: Customer Relationship Component instantiates Account Components
E-16: Welcome Mat instantiates Nav. Shell
E-17: Navigation Shell sends initial navigation choices to customer
E-18: Presentation Mgr formats style of screen display and sends response via Front Door Man etc. who do their normal functions



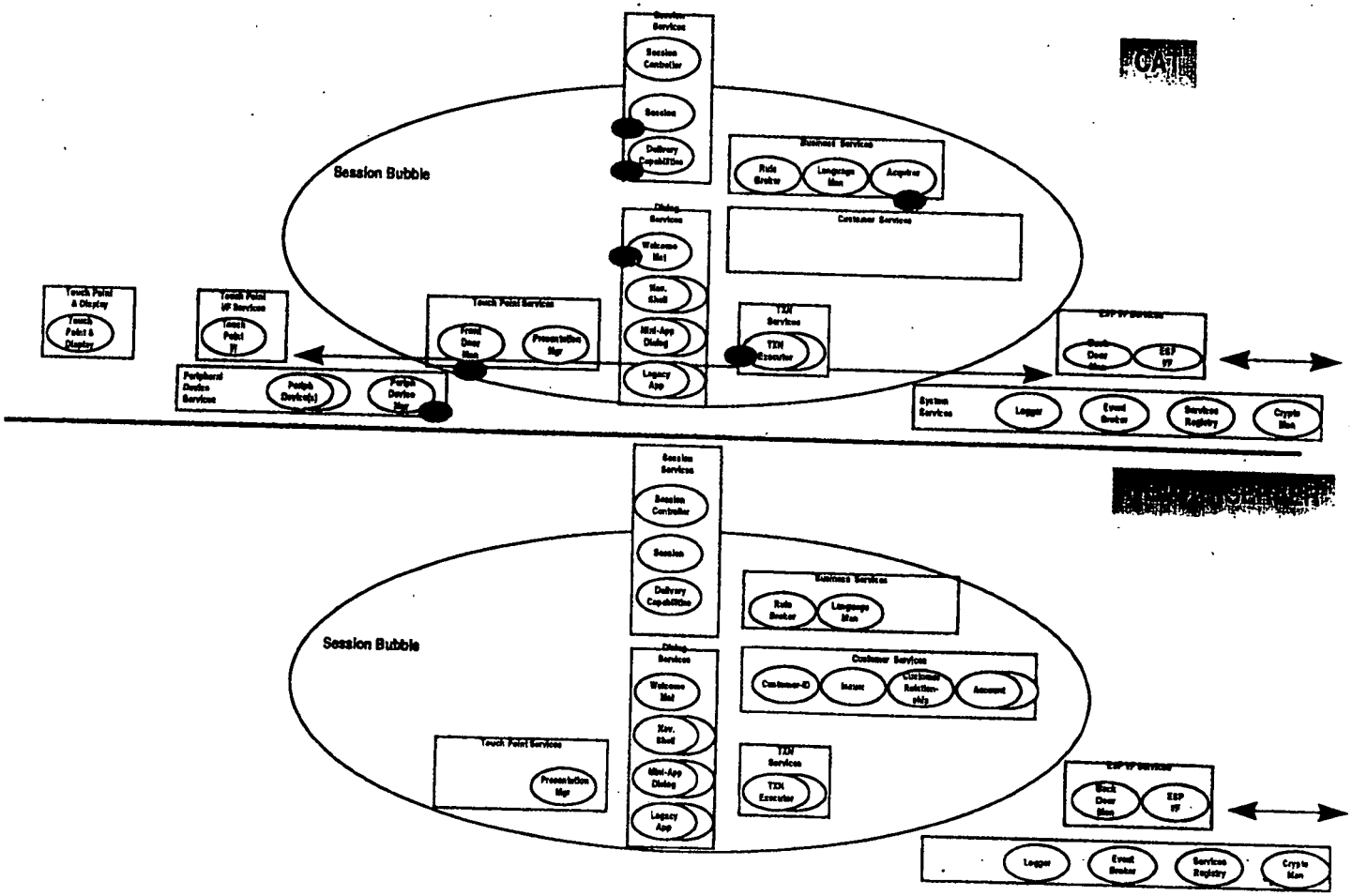
State 3 = Customer Picks Mini-App - Events 1 through 5



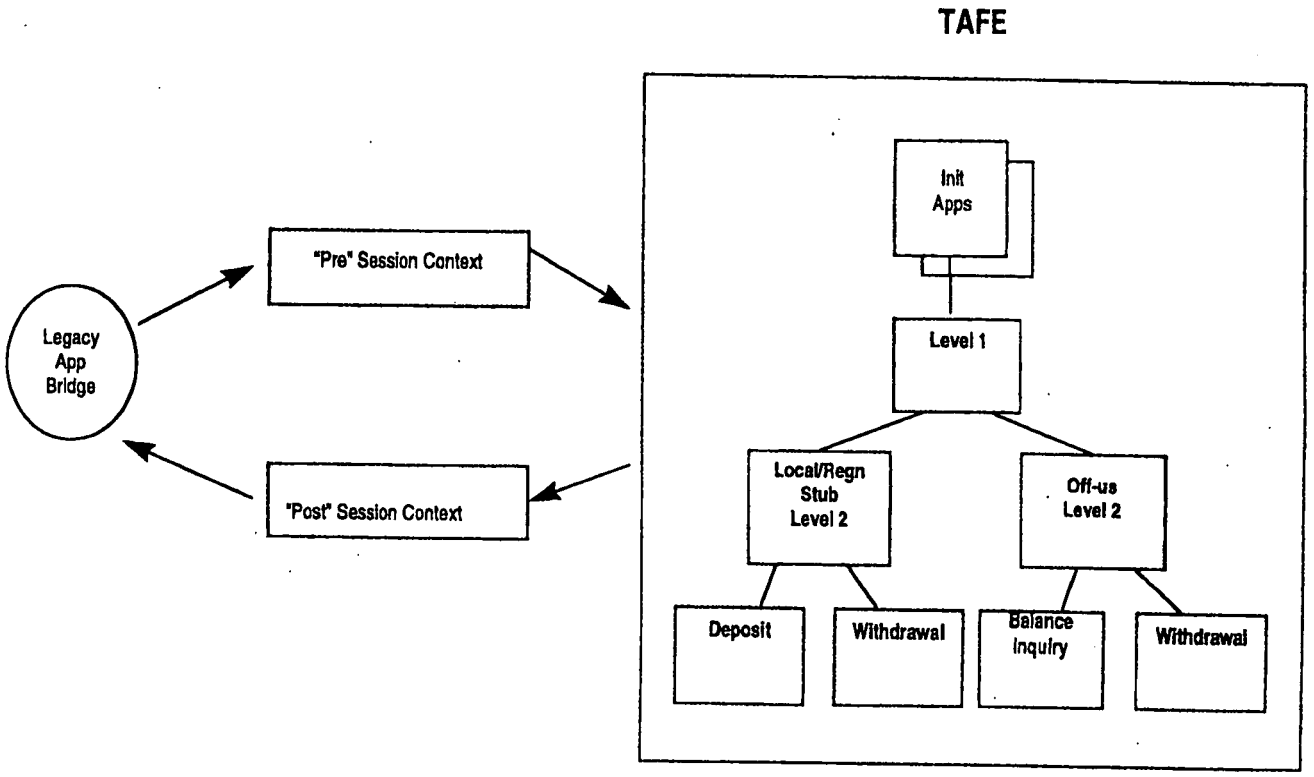
State 3 = Customer Picks Mini-App - Events 6 through 10



Net CAT Model



Interaction between CAT AGS applications and Service Components



Next Steps

- Obtain concurrence of proposed architecture
- Select development tools
- Define an implementation strategy
- Staff design team
- Start detail design